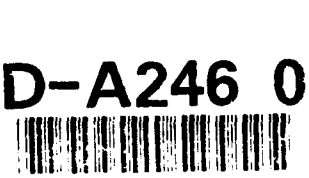


(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A PROTOTYPE SEMANTIC INTEGRITY FRONT END EXPERT SYSTEM FOR A RELATIONAL DATABASE

by

George Joseph Salitsky

September, 1991

Thesis Advisor: Magdi N. Kamel

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If applicable*) 37	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11. TITLE (*Include Security Classification*)
A PROTOTYPE SEMANTIC INTEGRITY FRONT END EXPERT SYSTEM FOR A RELATIONAL DATABASE

12. PERSONAL AUTHOR(S) Salitsky,George J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month, day*)	15. PAGE COUNT
Master's Thesis	From To	1991, September, 26	141

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Database integrity, Front end expert system, Prototype expert system, Semantic integrity.

19. ABSTRACT (*continue on reverse if necessary and identify by block number*)

Information is a critical resource in today's enterprises. Whether they are industrial, commercial, educational, or military, these organizations maintain an ever increasing amount of information in databases. Ensuring the accuracy of information in a database is paramount to the organization that maintain these databases. Many decisions are made from the information extracted from the database, and incorrect data will lead to incorrect decision making. This thesis examines the feasibility of using expert systems for enforcing semantic integrity constraints to relational databases. To accomplish this goal, the thesis develops a classification for semantic integrity constraints, applies it to develop rules for the Navy's Naval Aircraft Flight Record application, and builds a front end expert system to enforce these rules dynamically. The expert system enforces integrity rules for all maintenance operations(UPDATE, INSERT, and DELETE.)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Magdi N. Kamel	22b. TELEPHONE (*Include Area code*) (408)646-2494	22c. OFFICE SYMBOL AS/KA

Approved for public release; distribution is unlimited.

A Prototype Semantic Integrity
Front End Expert System
for a Relational Database

by

George J. Salitsky
Lieutenant, United States Navy
B.S., University of Scranton

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September, 1991

Author: _____
George J. Salitsky

Approved by: _____
Magdi N. Kamel, Thesis Advisor

_____
Hemant Bhargava, Second Reader

_____
David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

Information is a critical resource in today's enterprises. Whether they are industrial, commercial, educational, or military, these organizations maintain an ever increasing amount of information in databases. Ensuring the accuracy of information in a database is paramount to the organizations that maintain these databases. Many decisions are made from the information extracted from the database, and incorrect data will lead to incorrect decision making.

This thesis examines the feasibility of using expert systems for enforcing semantic integrity constraints to relational databases. To accomplish this goal, the thesis develops a classification for semantic integrity constraints, applies it to develop rules for the Navy's Naval Aircraft Flight Record application, and builds a front end expert system to enforce these rules dynamically. The expert system enforces integrity rules for all maintenance operations (UPDATE, INSERT, and DELETE.)

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Semantic integrity is concerned with ensuring that the database is always in a correct state even though some users or application programs may attempt to change it to an incorrect state. Enforcing semantic integrity means shielding the database against invalid UPDATES, INSERTS, and DELETIONS. Traditionally, most integrity checks are performed by application programs or by periodic auditing of the database. Problems of relying on application programs for integrity checks include:

- Application programs that modify the database could corrupt the whole database. That is, integrity checking is likely to be incomplete because the application programmer may not be aware of the semantics of the complete database.
- The criteria for integrity are buried in procedures and are therefore hard to understand and control.
- Code to enforce the same integrity constraints occurs in any number of applications; therefore inconsistencies could be introduced easily.

Problems of these types could be detected through the use of periodic auditing. Periodic auditing, on the other hand, causes problems because of the time lag in detecting errors. These problems include:

- There is considerable difficulty in tracing the source of an error and correcting it.

- The incorrect data may have been used to propagate other errors within the database and ultimately lead to incorrect decisions based on incorrect data.

Thus the prevention of inaccurate data into the database rather than the repair of the database once the damage has occurred is the preferred method. The enforcement of these integrity rules should be the responsibility of the DBMS, but DBMS vendors have failed to provide adequate integrity features to ensure accurate data within the database. [Ref. 1: p.109]

B. OBJECTIVES

This thesis suggests the use of a front-end expert system to enforce semantic integrity features. This expert system would oversee the update, insertion, and deletion operations, monitoring for violations of integrity rules. Once a violation had been identified, the system would take an appropriate action. This appropriate action would mean rejecting the operation and reporting the violation.

To understand how this will be accomplished, consider Figure 1. The expert system has a set of integrity rules that define what errors will be checked. These rules are stored in a knowledge base, which the inference engine of the expert system uses to enforce database integrity. The major advantage of this approach is that the validation of all data is handled by the expert system, instead of being left to the user or the

application program. Another important advantage is that all the integrity rules are located in the expert system's knowledge base. With the knowledge base acting as a central library, each integrity rule is easily queried and can be changed as needed.

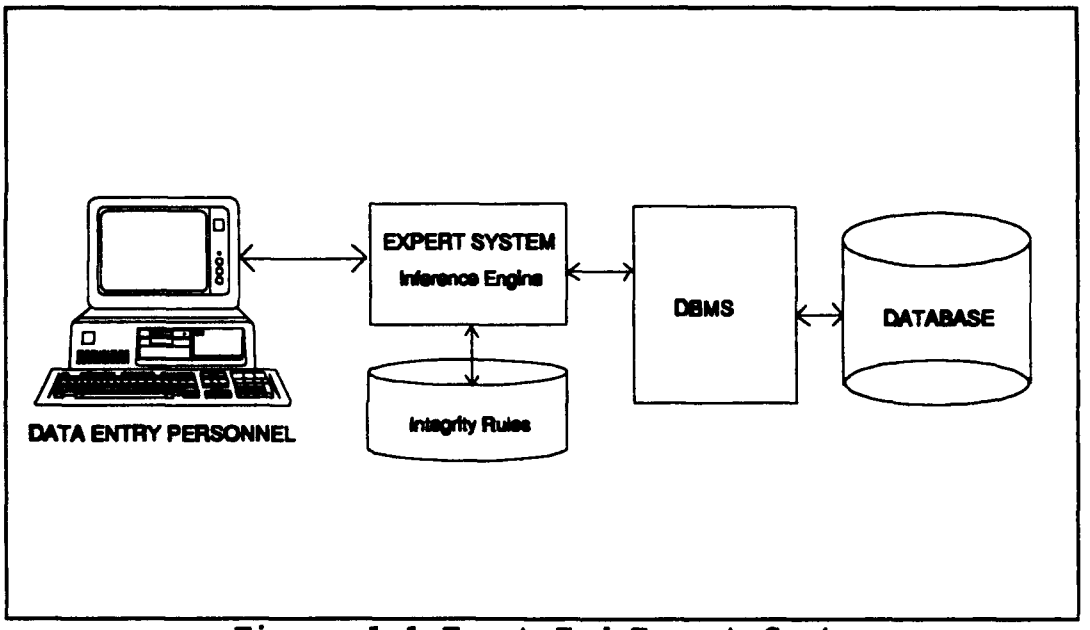


Figure 1.1 Front End Expert System

C. RESEARCH QUESTIONS

The Navy, through the use of the Naval Aircraft Flight Record, collects data for the Individual Flight Activity Reporting System(IFARS). The IFARS is a data bank for information that the Navy uses for safety analysis, budget justification of hours flown, and pilot compliance of established minimum standards. The accurate collection of data enables Naval Aviation to justify its existence while

providing the means to make it inherently less dangerous. The following research questions will be addressed:

- What are the integrity constraints to be enforced by a front end expert system based on the Navy's Naval Aircraft Flight Record, OPNAV 3710/4 and how will these constraints be classified?
- What is the feasibility of using an expert system as a front end in developing and enforcing these integrity constraints in a relational database application such as the Navy's Naval Aircraft Flight Record?

D. SCOPE

This thesis develops a semantic integrity front end expert system that monitors maintenance operations to a relational database developed for the Navy's Naval Aircraft Flight Record. It will address the issue of classification of integrity constraints to provide a structure for the knowledge base. It will also design a relational database representative of the way the user perceives the data on the Naval Aircraft Flight Record. Lastly it will design and implement a prototype front end expert system to enforce the integrity constraints developed, and maintain semantic integrity on the database. This prototype will be limited in its' ability to capture all data required by the Naval Aircraft Flight Record. It was not feasible to include all data or integrity constraints related to the data in the Naval Aircraft Flight Record due to the time constraint on this thesis.

E. ORGANIZATION OF THE STUDY

The thesis is organized as follows. Chapter II provides a classification of the integrity constraints that need to be incorporated into the expert system. Chapter III addresses the design of the relational database for the Naval Aircraft Flight Record application and describes the integrity rules that need to be enforced for this application. Chapter IV describes the design and construction of the front end expert system. Chapter V presents the conclusions of the research, as well as the benefits, limitations, and weaknesses of using a front end expert system.

II. CLASSIFICATION OF INTEGRITY CONSTRAINTS

An important goal of any database system is to model the real world accurately, and in a manner consistent with the user's perception of the data. The relational database model is based on the abstraction that data is stored in two-dimensional tables called relations. Each row in the table represents a tuple and each column represents an attribute. The entire table is equivalent to a file with all the properties of that relation. One of the fundamental principles of the relational database model is that relationships among distinct relations are captured through common values. Certain restrictions must be imposed on these relations to insure the integrity of the data within the database and allow for meaningful comparisons. The following is a list of integrity constraints that must be incorporated into the relational database model to guarantee these meaningful comparisons [Ref. 2].

- Domain Integrity Constraints
- Column Integrity Constraints
- Entity Integrity Constraints
- Referential Integrity Constraints
- User-Defined Integrity Constraints

Each type of constraint is detailed in the following sections.

A. Domain Integrity Constraints

The domain is the fundamental concept of the relational database model. The domain is the set of all possible values an attribute can have. It includes a physical description of:

- the data type
- range of values permitted for all columns within that domain
- allowable comparison operators (e.g., greater than (>) and less than (<))

and a semantic description (the function or purpose of the variable). A pair of values can be meaningfully compared, if and only if these values are drawn from a common domain. Consider the Naval Aircraft Flight Record in Figure 2.1.

DOC#	Document Number
SIDE#	Aircraft Side Number
EXCD	Exception Code
BUNO#	Aircraft Serial Number
ORG	Organization Code
MSN1	Mission Code
HRS1	Mission Hours
TOTFLT	Total Flights
ENG1	Engine 1 Hours
ENG2	Engine 2 Hours
ENG3	Engine 3 Hours
ENG4	Engine 4 Hours

DOC	EXCD	SIDE	BUNO	ORG	MSN1	HRS1	TOTFLT	ENG1	ENG2	ENG3	ENG4
001	C	052	152942	VP5	1A2	10.2	01	6.4	10.2	10.2	8.4

Figure 2.1 Domain Integrity Constraint

If both SIDE# and TOTFLT were declared to be numeric data type, a query to list all aircraft by SIDE#, where TOTFLT is greater than SIDE# would be a valid query. A query of this type would produce as much meaningful information as comparing

apples to oranges. Enforcing domain constraints ensures that two fields being compared not only have the same data types but also are semantically comparable. This feature safeguards users from meaningless information which could result from comparisons of values from different domains. Although special cases do arise that require the comparison of different domains, these should be exceptions and handled as such.

The use of domain constraints results in an integrated relational database[Ref. 2:p.45]. An advantage of this integration is logical value-comparisons. As can be imagined, the domain concept is fundamental to the support of each of the other integrity constraints that are mentioned. Domain constraints are what hold the relational database together and allow it to model the real world accurately and in conjunction with the user's way of thinking.

Today's DBMSs unfortunately do not support the domain concept. What they do support is basic data types(e.g., character, integer, float, calendar date, and clock times) and the ability to define certain ranges on these data types.

B. Column Integrity Constraints

Column integrity constraints are a natural extension of the domain concept. If the relational database supports the domain concept, then it should be capable of declaring in which domain the column belongs(inheriting the physical and semantic constraints associated with that domain), and any

additional constraints that are to apply to the columns. Each column name then becomes a combination of a role name and a domain name, where the role name designates the purpose of the column's use in a specified domain. The advantages are as follows:

- The description of every column that belongs to a given domain need only be declared once in the domain declaration.
- Because a given domain need only be declared once, the valid state of the database is ensured in future updates to integrity constraints.
- Support for ensuring database values are semantically comparable by checking to see if the columns belong to a common domain.
- Column integrity constraints are facilitated.

The last advantage is very important. If the relational database supports the domain concept, it has the ability to detect column integrity violations. Therefore, users can depend on the relational database to determine whether values in two different columns are semantically comparable.

Column integrity constraints may include the following:

- An added range constraint that provides a more confined range than in the domain declaration
- If missing values are allowed within a column
- Whether values must be distinct from each other within the column (primary keys)

Consider once again the Naval Aircraft Flight Record in Figure 2.1. HRS1, ENG1, ENG2, ENG3, and ENG4 belong to the same domain called Hours. The domain data type is a float type with

one decimal place. The range of values allowed is only positive. Negative values are not feasible. The column constraints for both HRS1 and ENG# are more restricted in that the range of values allowed is only between 00.1 to 72.0. Missing values are not allowed within the columns as long as the Exception Code is not X. ENG# value must be equal to or less than HRS1. This condition is specified to allow for engines that are shut down during a flight. Although some of these constraints within the example deal with other classes of integrity constraints, the basic idea of column integrity can be seen.

C. Entity Integrity Constraints

In order to understand Entity Integrity and Referential Integrity, it is important to discuss primary and foreign keys. Each row of a particular table in a relational database contains a column which contains primary-key values that uniquely identify and distinguish that row from every other row in that table. The primary-key can be composite and formed from more than one column. Everywhere else in the database that there is a need to refer to that unique row, the same value from the same domain is used but is referred to as a foreign-key value. The column that the foreign-key value is taken from is called the foreign key.

Entity Integrity implies that no component of a primary key is allowed to have a missing value. The primary-key in the

relational database model is a compulsory feature. An example of this is shown in Figure 2.2. The primary-key Document Number is missing from both records which is a violation of the Entity Integrity rule since it creates unidentified objects within the database. From Figure 2.3 we can see that duplicate primary-key values are prohibited, because of basically the same consequences (loss of identity).

Also, no component of a foreign key is allowed to be missing and inapplicable as opposed to missing and applicable. This case requires additional attention in that Side Number must adhere to referential integrity.

DOC	EXCD	SIDE	BUNO	ORG	MSN1	HRS1	TOTFLT	ENG1	ENG2	ENG3	ENG4
	C	052	152942	VP5	1A2	10.2	01	6.4	10.2	10.2	10.2
	C	052	152942	VP5	1A2	9.3	01	7.0	9.3	9.3	9.3

Figure 2.2 Entity Integrity Constraint(Missing)

DOC	EXCD	SIDE	BUNO	ORG	MSN1	HRS1	TOTFLT	ENG1	ENG2	ENG3	ENG4
001	C	052	152942	VP5	1A2	10.2	01	6.4	10.2	10.2	10.2
001	C	052	152942	VP5	1A2	9.3	01	7.0	9.3	9.3	9.3

Figure 2.3 Entity Integrity Constraint(Duplicate)

D. Referential Integrity Constraints

For each distinct foreign-key value in a relational database, there must exist in the database an equal value of a primary key from the same domain. If the foreign key is composite, those components that are themselves foreign keys must exist in the database as components of at least one

primary-key value drawn from the same domain. Consider the relational diagram in Figure 2.4. Aircraft Side Number is the primary-key value of the AIRCRAFT relation. Aircraft Side Number is also a foreign-key in the FLIGHT relation. From the relational diagram, FLIGHT must have one and only one Aircraft Side Number per document number while the relation AIRCRAFT can have one or more FLIGHTs associated with an Aircraft Side Number.

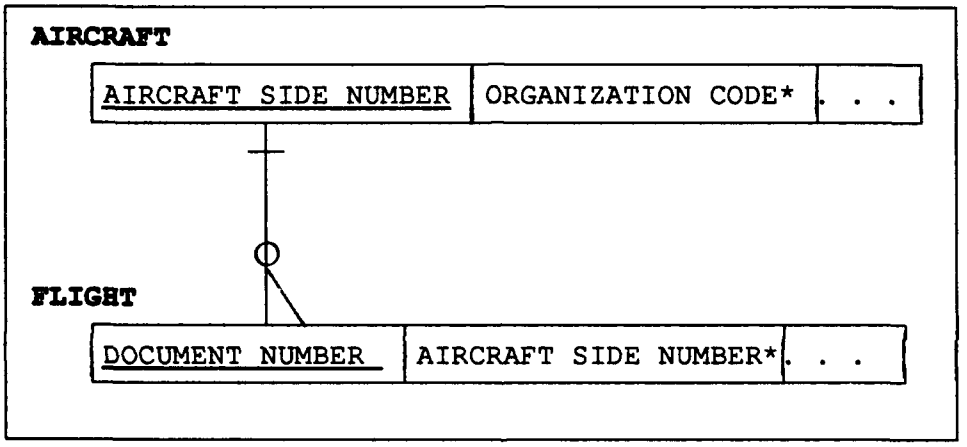


Figure 2.4 Relational Diagram

The entry of Document Number 0003AAA into the Flight relation in Figure 2.5 violates referential integrity because the Side Number 045 is not a primary-key in the Aircraft relation. Referential integrity can be thought of as inclusion dependency in that the foreign key must be a subset of a database in which it is the primary key.

E. User Defined Integrity Constraints

Domain, column, entity, and referential integrity are the building blocks of the relational database. User defined

integrity constraints are constraints that are peculiar to the end-user or company. These constraints allow organization practices and policy, or governmental legislation to be reflected in the database delineated by the user. Consider the Naval Aircraft Flight Record in Figure 2.6. The exception code

AIRCRAFT RELATION

SIDE#	Aircraft Side Number
ORG	Organization Code

SIDE#	ORG
051	VP5
052	VP5
053	VP5

FLIGHT RELATION

DOCNUM	Document Number
SIDE#	Aircraft Side Number

DOCNUM	SIDE#
0001AAA	052
0002AAA	051
0003AAA	045

Figure 2.5 Referential Integrity Constraint

DOC	EXCD	SIDE	BUNO	ORG	MSN1	HRS1	TOTFLT	ENG1	ENG2	ENG3	ENG4
001	X	052	152942	VP5	1A2	10.2	01	6.4	10.2	10.2	10.2

Figure 2.6 User Defined Integrity Constraint 1

X is used to document a canceled flight. A canceled flight is one for which no flight time is obtained. Document 001 has violated a user defined integrity rule because it has allowed flight time to be documented for a canceled flight.

User defined constraints such as this, require that UPDATE operations have an ordered sequence of events in order to

comply with all the integrity constraints defined for the database. Examine the Naval Aircraft Flight Record in Figure 2.7. In an UPDATE operation on Document 001 the Exception Code was changed to X. This resulted in the record change in the database demonstrated in Figure 2.8. Not only did all flight time need to be removed, the Mission Code needed to be changed to reflect the user defined constraint that the 2nd position of the Mission Code be N or the character O if the Exception Code is an X.

DOC	EXCD	SIDE	BUNO	ORG	MSN1	HRS1	TOTFLT	ENG1	ENG2	ENG3	ENG4
001	X	052	152942	VP5	1A2	10.2	01	6.4	10.2	10.2	10.2

Figure 2.7 User Defined Integrity Constraint 2

DOC	EXCD	SIDE	BUNO	ORG	MSN1	HRS1	TOTFLT	ENG1	ENG2	ENG3	ENG4
001	X	052	152942	VP5	1N2						

Figure 2.8 User Defined Integrity Constraint 3

The intent of this chapter has been to develop the framework for the expert system. Classifying the integrity constraints allows for the building of rules according to these constraints. In order for the expert system to function properly, the integrity constraints must be transparent to the user so that there is no reliance on voluntary action by the user to maintain integrity within the database. In regard to transparency, attempted violations of the integrity constraints must be denied with an appropriate reason for

denial conveyed to the user. Also, any operations on the database must be atomic in the sense that each operation must be completed satisfactorily (satisfying all integrity constraints) or denied and rolled back to its original state.

III. NAVAL AIRCRAFT FLIGHT RECORD RELATIONAL DATABASE DESIGN

As discussed in Chapter I, the thrust of this thesis is the feasibility of using a front end expert system to enforce semantic integrity constraints. This chapter discusses the development of a relational database model and its associated semantic integrity rules that will serve as the case study for the front end expert system.

A. BACKGROUND

The relational database model developed in this chapter is based on the Naval Aircraft Flight Record(OPNAV 3710/4), shown in Figure 3.1. This record serves as the sole source of all naval aircraft flight data and is applicable in specific areas to aircraft simulators. The OPNAV 3710/4 record is prepared for each attempt at flight of naval aircraft or training evolution for simulators. The types of data collected are:

- A statistical description of the flight pertaining to the aircraft and crew members
- A record of all logistic actions performed during the flight
- A record of weapons proficiency
- A record of training areas utilized and other miscellaneous data

The Operations Department within the aircraft squadron is responsible for verifying the accuracy and completeness of

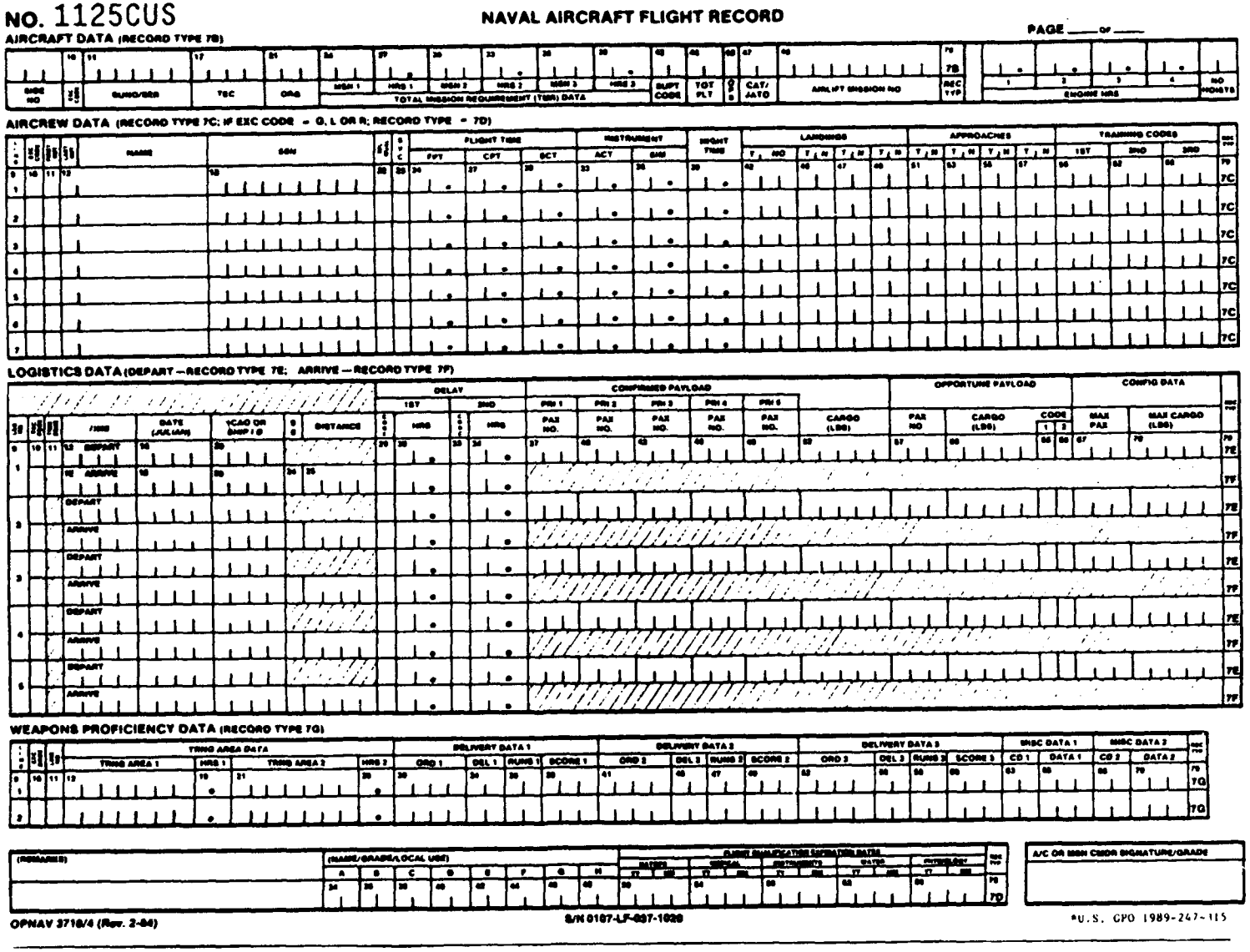
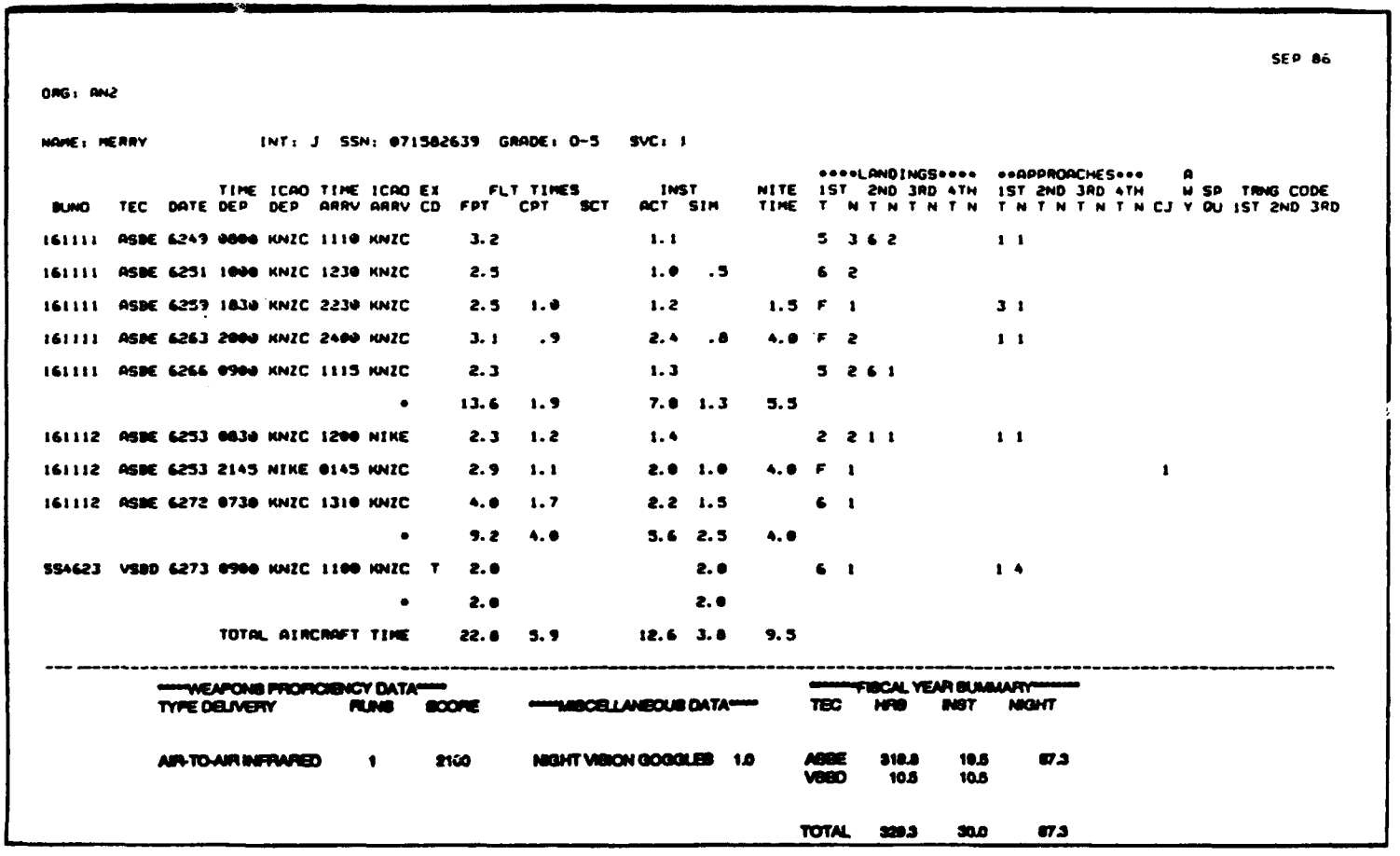


FIGURE 3.1 Naval Aircraft Flight Record (OPNAV3710/4)

naval aircraft flight records submitted for data processing as well as verifying the daily audit reports, and coordinating the correction of errors with the maintenance analyst. The Maintenance Analyst is the NAVFLIRS coordinator who is responsible for accomplishing the daily submission of completed naval aircraft flight records for processing, distributing daily audit and monthly reports to the operations and maintenance departments, and coordinating error corrections with operation and maintenance control. Completed naval aircraft flight records are then forwarded to the Naval Safety Center (NAVSAFCEN) for processing. A Monthly Individual Flight Activity Report(MIFAR), shown in Figure 3.2, is produced by the NAVFLIRS system and forwarded to the aviator by NAVSAFCEN. The MIFAR contains all individual activity for that month, excluding those records appearing on the error reports processed by NAVSAFCEN. This includes a summarization by aircraft bureau number and by the flight times (First Pilot Time(FPT), Co-Pilot Time(CPT), and Special Crew Time(SCT)), including instrument (Actual Instrument Time(ACT) and Simulated Instrument Time(SIM)), and night times for that month. The MIFAR also contains a weapons proficiency summary, a miscellaneous data section, and a fiscal year to date summary indicating what is on record in the NAVFLIRS system. In addition to producing the MIFAR, the NAVSAFCEN is the collection and maintenance activity for the IFARS data bank. The IFARS is the primary source of individual flight data,



SEP 86

ORG: AN2

NAME: MERRY INT: J SSN: 071582639 GRADE: O-5 SVC: 1

BUNO	TEC	DATE	TIME DEP	ICAO DEP	TIME ARRV	ICAO ARRV	EX CD	FLT TIMES FPT	CPT	SCT	INST ACT	SIM	NITE TIME	••••LANDINGS•••• 1ST T N	2ND T N	3RD T N	4TH T N	••APPROACHES••• 1ST T N	2ND T N	3RD T N	4TH T N	A W CJ	SP Y	QU	TRNG CODE 1ST	2ND	3RD
161111	ASBE	6249	0800	KNIC	1110	KNIC		3.2			1.1			5	3 6	2		1	1								
161111	ASBE	6251	1000	KNIC	1230	KNIC		2.5			1.0	.5		6	2												
161111	ASBE	6259	1830	KNIC	2230	KNIC		2.5	1.0		1.2		1.5	F	1			3	1								
161111	ASBE	6263	2000	KNIC	2400	KNIC		3.1	.9		2.4	.8	4.0	F	2			1	1								
161111	ASBE	6266	0900	KNIC	1115	KNIC		2.3			1.3			5	2 6	1											
						•		13.6	1.9		7.8	1.3	5.5														
161112	ASBE	6253	0830	KNIC	1200	NIKE		2.3	1.2		1.4			2	2 1	1		1	1								
161112	ASBE	6253	2145	NIKE	0145	KNIC		2.9	1.1		2.0	1.0	4.0	F	1							1					
161112	ASBE	6272	0730	KNIC	1310	KNIC		4.0	1.7		2.2	1.5		6	1												
						•		9.2	4.0		5.6	2.5	4.0														
554623	VSBD	6273	0900	KNIC	1100	KNIC	T	2.0				2.0		6	1			1	4								
						•		2.0				2.0															
		TOTAL AIRCRAFT TIME						22.8	5.9		12.6	3.8	9.5														

••••WEAPONS PROFICIENCY DATA••••

TYPE DELIVERY	RUNS	SCORE
AIR-TO-AIR INFRARED	1	2100

••••MISCELLANEOUS DATA••••

NIGHT VISION GOGGLES	1.0

••••••FISCAL YEAR SUMMARY••••••

TEC	HRS	INST	NIGHT
ASBE	318.8	19.5	87.3
VSBD	10.5	10.5	
TOTAL	329.3	30.0	87.3

Figure 3.2 Monthly Individual Flight Activity Report

including those flights flown in authorized simulators. The reporting vehicle for IFARS data is the Naval Aircraft Flight Record OPNAV 3710/4. The IFARS data bank provides valuable exposure data for flight safety analysis and also provides data for other uses such as budget justification, past and future program evaluation, and pilot compliance with established minimum standards. Commander of Naval Military Personnel Command(COMNAVMILPERSCOM) annually convenes a flight board to review pilot flight activity by looking at the IFARS data bank against the annual flying requirements as set forth in OPNAVINST 3710. Each year, the Naval Safety Center mails to reporting individuals their flight data report for the previous fiscal year. IFARS data is applicable to naval aviators, student naval aviators, naval flight officers, aviation pilots flying naval aircraft, naval flight surgeons, and aerospace physiologists/psychologists in a DIFOPS(duty in a flying status for an officer involving operational or training flights) or DIFDEN(duty in a flying status for an officer not involving flying) status on active duty or participating in the Navy or Marine reserve program. [Ref. 3:pp.10(1-4)]

B. AIRCRAFT FLIGHT RECORD OBJECTS

In order to develop a relational schema for the Naval Flight Data application, a series of objects were developed to capture the data requirements for the Naval Aircraft Flight

Record, OPNAV 3710/4. An object is a named collection of properties that sufficiently describes an entity in the user's work environment [Ref. 4:p.90]. The objects developed for this application include: ORGANIZATION, AIRCRAFT, AIRCREW, FLIGHT, AIRCREW-FLIGHT, LOGISTICS, ARRIVAL, DEPARTURE. In the following sections, each object is described in more detail. The complete Object Diagrams are shown in Appendix A.

1. ORGANIZATION Object

This object represents a generic naval aircraft squadron. It is identified by an Organization Code and includes properties such as Data Processing Code, Organization Short Name, Support Code, Departure Time Zone, Departure IACO, Cats/Jato, Airlift Mission, Payload Configuration Data, and Training Codes. Typically an organization will have several aircraft.

2. AIRCRAFT Object

This compound object represents a generic naval aircraft. It can be identified by the Aircraft Side Number or Buno/Serial Number and includes properties such as Type Equipment Code, and Number of Engines. Typically an aircraft is assigned to exactly one organization and is used for many flights.

3. FLIGHT Object

This compound object represents a generic naval aircraft flight. It is identified by the Document Number and

includes properties such as Exception Code, Total Flights, Ship/Field Operations Code, Catapult/Jato Launches, Airlift Mission Number, Number of Hoists, and Remarks. Mission Code, Mission Hours, Engine Number and Engine Hours are multi-valued properties and can contain more than single values. A flight can only involve one aircraft but may typically involve many aircrew members while carrying out many logistic missions.

4. AIRCREW Object

This object represents a generic naval aircrew member. It is identified by the Social Security Number and includes properties such as Last Name, First Initial, Service, Grade, Organization, Natops Qualification Expiration Date, Medical Expiration Date, Instrument Qualification Expiration Date, Water Qualification Expiration Date, Physiology Qualification Expiration Date, Assigned Syllabus, Syllabus Status Code, Aircrew Status Code, and Exception Code. Typically an aircrew member will be involved in many aircrew flights.

5. AIRCREW FLIGHT Object

This association object represents a generic naval aircrew flight. It is identified by the combination of properties, Document Number and Social Security Number. The justification for making AIRCREW FLIGHT an association object instead of a compound object stems from the fact that AIRCREW FLIGHT is perceived as an independent object. Independent, because it contains non-key data and documents a relation

between FLIGHT and AIRCREW. Its properties include First Pilot Time, Co-Pilot Time, Special Crew Time, Actual Instrument Time, Simulated Instrument Time, and Night Time. Multi-valued properties include Type Landings, Number Landings, Type Approach, Number Approaches, Training Code, Training Area, Training Hours, Ordnance Code, Delivery Code, Runs, Score, Miscellaneous Data Code and Miscellaneous Data.

6. LOGISTICS Object

This object represents a generic naval logistic flight leg. It is a composite object that is identified by the composite key of Document Number and Leg Number and contains the property Time Zone. Each logistic leg will be associated with a flight and have one arrival and departure.

7. DEPARTURE Object

This composite object represents a generic naval flight departure leg. It is identified by Document Number, Leg Number, and Departure Time. Its properties include Departure Date, Departure ICAO, Confirmed Payload Cargo, Opportune Payload Cargo, Maximum Passenger, and Maximum Cargo. Delay Departure Code, Delay Departure Hours, Passenger Priority, and Opportune Payload Code are multi-valued properties. Each departure will be associated with one logistic leg.

8. ARRIVAL Object

This composite object represents a generic naval flight arrival leg. It is identified by Document Number, Leg

Number, and Arrival Time. Its properties include Arrival Date, Arrival ICAO, System Status, and Distance. Delay Arrival Code and Delay Arrival Hours are multi-valued properties. Each arrival will be associated with one logistic leg.

C. NAVAL AIRCRAFT FLIGHT RECORD SCHEMA

In this section we perform a logical database design by transforming the objects developed in the previous section into a relational schema. The output from this phase is a set of relations, relation definitions, relationships between relations, and constraints on these relationships. In the following sections, we discuss the main relations and relationships of the schema. Refer to the Object Diagrams in Appendix A and the Relational Diagrams in Appendix C for the following discussion.

1. ORGANIZATION Relation

This relation is transformed from the object ORGANIZATION. It is identified by the attribute organization code. This relation is associated in a one to many optional relationship with the AIRCRAFT relation. In other words, a record of this relation may be associated with one or more records of the AIRCRAFT relation.

2. AIRCRAFT Relation

This relation is transformed from the compound object AIRCRAFT. It is identified by the attribute aircraft side number. It contains the foreign attribute of organization code

from the ORGANIZATION relation. Whereas the ORGANIZATION did not need any instances of aircraft, the AIRCRAFT has a mandatory relationship with the ORGANIZATION. This represents a many to one mandatory relationship. On the other hand, the AIRCRAFT relation is associated in a one to many optional relation with the FLIGHT relation. As with the ORGANIZATION relation, a record in this relation may be associated with one or more records of the FLIGHT relation.

3. FLIGHT Relation

This relation is transformed from the compound object FLIGHT. It is identified by the attribute document side number. It contains the foreign attribute aircraft side number from the AIRCRAFT relation. FLIGHT is represented by a many to one mandatory relationship with AIRCRAFT, indicating that any records from this relation must be associated with one record of the parent AIRCRAFT. The object FLIGHT is also a composite object meaning that it contains repeating groups of non object properties. Each of these groups is represented by a relation in the database. The first relation, MISSION, is identified by the composite key document number and mission code. It is represented as a many to one mandatory relationship indicating the possibility of many mission records, each associated with a FLIGHT record. The second relation, ENGINE, is identified by document number and engine number. It is also represented as a many to one mandatory relationship, indicating as many

records as the aircraft has engines.

The relation FLIGHT also serves as the parent to both the relations AIRCREW FLIGHT and LOGISTICS. In both instances, the relation is associated in a one to many optional relationship. Each record of FLIGHT may be associated with one or more records of both the AIRCREW FLIGHT and LOGISTICS relations.

4. AIRCREW Relation

This relation is transformed from the object AIRCREW. It is identified by the attribute ssn (Social Security Number). This relation is associated in a one to many optional relationship with the AIRCREW FLIGHT relation. In other words, a record of this relation may be associated with one or more records of the AIRCREW FLIGHT relation.

5. AIRCREW FLIGHT Relation

This relation is transformed from the association object AIRCREW FLIGHT representing the relationship between FLIGHT and AIRCREW. The relation is identified by the composite properties of document number and ssn, each of which are the keys of the parent relations. Although this object does not contain a key of its own, it does contain non-key data that indicate details of a specific flight and represents a real object in the user's environment. The non-key data are represented by multiple repeating groups. Each of these repeating groups is represented by a relation with a one to

many optional relationship with AIRCREW FLIGHT. The first relation, LANDING, is identified by document number, ssn, and type landing. The second relation, APPROACH, is identified by document number, ssn, and type approach. The third relation, TRAINING, is identified by document number, ssn, and training code. The fourth relation, TRAINING AREA, is identified by document number, ssn, and training area. The fifth relation, WEAPONS, is identified by document number, ssn, and delivery number. The final relation within the association object is MISCELLANEOUS, identified by document number, ssn, and miscellaneous data code.

6. LOGISTICS Relation

This relation is transformed from the composite object LOGISTICS. It is identified by the composite properties document number and leg number. It is associated with FLIGHT in a many to one mandatory relationship indicating that any records in this relation must be associated with a record in the FLIGHT relation. The relation is also associated with the relations DEPARTURE and ARRIVAL as a one to one mandatory relation. Both relations DEPARTURE and ARRIVAL contain records that describe different aspects of the same relation LOGISTIC. Although these relations may be combined into one, a better user understanding of the relational database design and better database performance can be achieved by the separating the two.

7. DEPARTURE Relation

This relation is transformed from the composite object DEPARTURE. It is identified by the composite properties document number, leg number, and departure time. As was mentioned previously, it is represented as a one to one mandatory relationship with the LOGISTIC relation. It also contains multiple repeating groups represented by the following relations which maintain a one to many optional relationships with DEPARTURE. The first relation, PASSENGER, is identified by document number, leg number, and passenger priority. The second relation, PAYLOAD, is identified by document number, leg number, and opportune payload code. The last relation, DEPARTURE DELAY, is identified by document number, leg number, and delay departure code.

8. ARRIVAL Relation

This relation is transformed from the composite object ARRIVAL. It is identified by the composite properties document number, leg number, and arrival time. Once again, it is represented as a one to one mandatory relationship with the LOGISTIC relation. It is also represented by a relation, ARRIVAL DELAY, representing a one to many optional relationship. The relation is identified by document number, leg number, and delay arrival code.

D. INTEGRITY CONSTRAINTS

In this section, we present the semantic integrity rules that need to be maintained for the relational schema developed in the previous section [Ref. 5]. Due to the sheer size of the database design, it was decided to narrow the focus of the front end expert system by limiting the integrity constraints to the FLIGHT relation. The narrowed focus still allowed the system to address all the classes of integrity constraints developed in Chapter II.

1. Domain Integrity Constraints

The domain constraints enforced in this application are presented in Appendix B.

2. Column Integrity Constraints

The column constraints as discussed previously in Chapter II can be thought of as a subset of the domain integrity constraints. The following column integrity constraints are enforced in the front end expert system:

- Exception Code must be C, D, X, or BLANK
- Mission Code (n) where n = 1 must be in the range of 1-6 or BLANK
- Mission Code (n) where n > 1 must be in the range of 1-5 or BLANK
- Mission Hours (n) where n = 1 must be in the range of 0.1 to 72.0 or BLANK
- Mission Hours (n) where n > 1 must be in the range of 0.1 to (72.0 - Sum of Mission Hours) or BLANK
- Total Flight must be in the range of 1-99 or BLANK

- Ship/Field Operations must be A, B, 1, 2, or BLANK
- Catapult/Jato Launches must be in the range of 1-99 or BLANK
- Engine Hours (n, n+1, n+2,...) must be in the range of 0.1 to 72.0 or BLANK
- Number of Hoists must be in the range of 1-99 or BLANK

3. Entity Integrity Constraints

The following entity integrity constraints are enforced by the front end expert system:

- Document Number cannot be missing or duplicated
- Aircraft Side Number cannot be missing
- Mission Code (n) where n = 1 cannot be missing

4. Referential Integrity Constraints

The following referential integrity constraints are enforced by the front end expert system:

- Aircraft Side Number must be validated against the AIRCRAFT object for the purpose of recording a valid Buno/Serial number and ensuring the correct number of engines are recorded for flight time
- Document Number for the composite objects is the same as the FLIGHT document number

5. User Defined Integrity Constraints

The following user defined integrity constraints are enforced by the front end expert system.

a. Intra-Attribute Constraints

These user defined integrity constraints apply to the relationships within an attribute:

- Mission Code (n), Position 2, when n=1 or >1, must be R or in the range of A-I or N-P if Position 1 is a 1

- Mission Code (n), Position 2, when n=1 or >1,must be in the range of J-R if Position 1 is a 2
- Mission Code (n), Position 2, when n=1 must be O or in the range of S-Z if Position 1 is 3-6 or Position Code is 3-5 when n>1
- Mission Code (n), Position 2, when n=1 must be O or N if Exception Code is X
- Mission Code (n), Position 1,2, and 3 when n>1 must be BLANK when Exception Code is X
- Mission Code (n), Position 1,2, and 3 when n>2 must be BLANK when Mission Code (n-1) is BLANK

b. Intra-Relation Constraints

These user defined integrity constraints apply to the relationships within a relation:

- Mission Hours (n), when n=1 or >1, must be Blank if Exception Code is X
- The sum of Mission Hours (n+(n+1)+(n+2)+...) must not exceed 72.0 hours
- Mission Hours (n), when n>1, must be BLANK if Mission Code (n) is BLANK
- Total Flight must be BLANK if Exception Code is X
- Total Flight must meet its column integrity constraints if the Exception Code is not X
- Ship/Field Operations Code must be BLANK if Exception Code is X
- Ship/Field Operations Code must meet its column integrity constraints if the Exception Code is not X
- Catapult/Jato Launches must be BLANK if Exception Code is X
- Catapult/Jato Launches must meet its column integrity constraints if the Exception Code is not X
- Airlift Mission Number must be BLANK if Exception Code is X

- Airlift Mission Number must meet its column integrity constraints if the Exception Code is not X
- Engine Hours (n,n+1,n+2,...) must be BLANK if Exception Code is X
- Engine Hours (n,n+1,n+2,...) must be in the range of 0.1 to Mission Hours (n,n+1,n+2,...) if the Exception Code is not X
- Number of Hoists must be BLANK if Exception Code is X
- Number of Hoists must meet its column integrity constraints if the Exception Code is not X

In the next chapter, the design and implementation of a front end expert system that enforces the above integrity rules is described.

IV. DESIGN AND IMPLEMENTATION OF THE FRONT END EXPERT SYSTEM

Expert systems are programs that respond to information very much like a human expert in a well-defined area(the program's domain). They capture and distribute knowledge to the non-experts and general practitioners in specific application areas where:

- Difference in performance is largely based on expert knowledge.
- This knowledge is experienced-based.
- The knowledge can be stated as "If...then" rules [Ref. 6:p.17]

An important aspect of some expert systems is the ability to capture knowledge and then record it as a set of rules in a knowledge base. Expert system shells such as VP-Expert use an inference engine that interacts with the user and navigates through the knowledge base to deliver this knowledge.

A. INFERENCE ENGINE

The search strategy or problem solving method used in this thesis application and supported by VP-Expert is called "backward-chaining." The inference engine starts by identifying a target variable and then moves through a sequence of rules until it finds a value that can be assigned to that target variable. Consider the following example in

Figure 4.1.

In this example, any of the three rules can assign a value to TOTFLT_VALID. If the value for EXCD is not known then the inference engine looks for the rule assigning a value to EXCD

```
FIND TOTFLT_VALID;                          -The target variable is
                                             identified as
                                             TOTAL_FLIGHT_VALID

RULE USER_DEFINED_CONSTRAINT_TOTFLT_1
IF
     EXCD = X                               -If Exception Code is
                                             equal to the value "X"
THEN
     TOTFLT = (BLANK)                       -Then assign a null
     TOTFLT_VALID = TRUE;                    value to TOTFLT and
                                             assign TRUE to
                                             TOTAL_VALID

RULE USER_DEFINED_CONSTRAINT_TOTFLT_2
IF
     EXCD <> X AND                          -If Exception Code is
     TOTFLT >= 1 AND                         not equal "X" and the
     TOTFLT <= 99                            value assigned to
                                             TOTFLT is greater than
                                             0 and less than 100
THEN
     TOTFLT_VALID = TRUE;                   -Then assign TRUE to
                                             TOTFLT_VALID

RULE USER_DEFINED_CONSTRAINT_TOTFLT_3
IF
     EXCD <> X AND                          -If Exception Code is
     TOTFLT < 1 OR                           not equal "X" and the
     TOTFLT > 99                             value assigned to
                                             TOTFLT is less than
                                             1 or greater than 99
THEN
     TOTFLT_VALID = FALSE;                  -Then assign FALSE to
                                             TOTFLT_VALID
```

Figure 4.1 "Backward" Chaining

in its conclusion. If the value assigned to EXCD is X,USER_DEFINED_CONSTRAINT_TOTFLT_1 is fired and the value for

TOTFLT becomes null. On the other hand if the value of EXCD is not equal to X, then the first rule is passed and the second rule is applied. Once again, if the value for TOTFLT is not known, then the inference engine must look for a rule that assigns a value to TOTFLT. This pattern continues if other variables within the rule were not known. Once all the values are known, the inference engine retraces its steps and tests the original rule. In the example above if TOTFLT is 2, then rule USER_DEFINED_CONSTRAINT_TOTFLT_2 is fired and TOTFLT_VALID is assigned TRUE.

B. APPLICATION DESIGN

The front end expert system is the user's interface with the database. It is designed to perform maintenance on the database to include append, update, and delete operations. While the rules have been defined in the last chapter, this section deals with the logic needed in the application. Which questions are asked initially? Which answers lead to other questions? In the following sections we discuss each of the maintenance operations.

NOTE: While all the maintenance operations require access to all objects of the database design, no maintenance operations are allowed on the following objects; ORGANIZATION, AIRCRAFT, and AIRCREW. The security of these objects require that they be protected from either malicious or accidental destruction or corruption.

1. Append

After the user selects APPEND RECORD from the main menu, the expert system uses a system-generated dialogue with the user to generate a record for the FLIGHT object. Each attribute is checked against the integrity constraints for that specific attribute by the inference engine. Each attribute that meets the constraints imposed by the expert system is stored until the end of the transaction. If the attribute cannot meet the integrity constraints of the knowledge base, the system continues to ask the user for the attribute and offers assistance as to a valid attribute the system will accept. This feature disallows an invalid attribute and prevents the invalid record from being added to the database, since the user cannot continue until a valid attribute is entered.

The logical ordering of questions follow from the Naval Aircraft Flight Record(OPNAV 3710/4)as shown in Figure 3.1. Some of the answers that lead to other questions include the following:

- Exception Code = X
- Mission Code 1/Position 1 = 6
- Mission Code 2 = Unknown

These answers affect the logical ordering of questions to be asked. The rules from Figure 4.1 used earlier in finding TOTFLT_VALID show this ordering. If the Exception Code is

equal to X then TOTFLT is set to null. This made the attribute TOTFLT appear to be overlooked, when in fact the rule USER_DEFINED_CONSTRAINT_TOTFLT_1 fired and assigned (BLANK) to the attribute TOTFLT.

At the end of the append operation, all values assigned to the attributes are committed to the database. If at any time during the transaction the user quits, or the append operation is terminated, the attribute values are effectively rolled back to their previous values.

2. Update

This maintenance operation is probably the most critical of all the operations. Questions, that are asked in a logical order in the append operation, may not have been asked when updating the value of one attribute. The ability to change attribute values of a record requires a clear understanding of the semantics of the whole database.

The selection of the UPDATE RECORD from the main menu provides the user with another menu showing all possible Naval Aircraft Flight Records within the database to update. After selection of a record, the user is then presented with a sub-menu of all possible attributes to update. The changing of one attribute may not only fire the rule for that attribute but may also fire multiple other rules for attributes that are logically affected by the update of that attribute. For example, a Naval Aircraft Flight Record with the attribute

Exception Code equal to X designates a canceled flight and, therefore, cannot contain flight data. In the event that the canceled flight was later flown, an update to the record should ensure that all the attributes of a flight are updated. Because of this, each unique update operation fires a separate rule. This presents a logical ordering of questions, which preserves the semantic integrity of the record in conjunction with the attribute updated. Figure 4.2 is an example of one of many update rules searched to update Mission Code 1. The inference engine searches the knowledge base after a valid Mission Code 1 has been entered to provide the logic that is needed to preserve the integrity of the record. This rule could only fire after Mission Code 1 met the Integrity Constraints defined in Chapter III. No attributes are committed to the database until all attributes meet all integrity constraints as determined by the inference engine.

3. Delete

The final maintenance operation deals with purging the database of unwanted records. This requires a cascaded delete operation. This operation deletes the designated FLIGHT record and all optional records related to the deleted FLIGHT record. These relations are shown in the relational schema of Appendix C. This function is based on referential integrity and the associated concept of inclusion dependency as discussed in Chapter II. Because this operation is potentially destructive,

a confirmation message that explains the consequences of the process is displayed, and the user is given the opportunity to cancel the delete operation. This operation doesn't mark the

```
RULE MISSION_CODE_1_RULE_2
IF
 FIELD_TO_UPDATE = MISSION_CODE_1 AND  -Field to Update =
 MSN1_1 = 6 AND                         Mission Code 1 and
 EXCD <> X                             -Mission Code 1
                                        Position 1 = 6 and
                                       -Exception Code = "X"
THEN
 MISSION_CODE1_RULE = USED             -Assigns USED to
 TOTAL = 0                              target variable
 FIND HRS1_VALID                       -Assigns 0 to TOTAL
 MSN2_1 = (BLANK)                       for Total Hours flown
 MSN2_2 = (BLANK)                      -Looks for HRS1_VALID
 MSN2_3 = (BLANK)                      -Assigns null to
 HRS2 = (BLANK)                         MSN2_1
 MSN3_1 = (BLANK)                      -Assigns null to
 MSN3_2 = (BLANK)                       MSN2_2
 MSN3_3 = (BLANK)                      -Assigns null to
 HRS3 = (BLANK)                         MSN2_3
 FIND MISSION1_ENGHRS_VALID            -Assigns null to
 PUT FLIGHT                             HRS2
 CLOSE FLIGHT;                         -Assigns null to
                                        MSN3_1
                                       -Assigns null to
                                        MSN3_2
                                       -Assigns null to
                                        MSN3_3
                                       -Assigns null to
                                        HRS3
                                       -Looks for
                                        MISSION1_ENGHRS_VALID
                                       -Commits attributes to
                                        database
                                       -Closes database
```

Figure 4.2 Update Operation Mission Code 1

record for deletion, instead it assigns an unknown value (BLANK) to each attribute of the record and then commits these values to the associated relations.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This thesis has addressed the issue of dynamic enforcement of integrity constraints in a relational database through the use of a front end expert system. It has also addressed the classification of integrity constraints as a framework for designing and building the front end expert system. The development of a front end expert system for the Navy's Naval Aircraft Flight Record served as the vehicle for demonstrating the feasibility of this concept in a well-defined, structured area.

Although limited in functionality, the Naval Aircraft Flight Record front end expert system was successful in maintaining semantic integrity for any given maintenance operation(insertion, deletion, and update.) Because of the atomic nature of all maintenance operations, the integrity of the database is guaranteed at all times. A separate validation program is, therefore, not required to audit the database periodically.

The use of an expert shell with an If...Then construct proved to be a viable method to test and implement the integrity constraints developed. The ability to store these rules in one central repository (knowledge base) was the most

significant benefit of using an expert shell. Any maintenance to the program itself was made easier by the ability of the user to ask why a particular response was obtained. This allowed for query of the appropriate rule and examination of the constraints imposed, therefore simplifying program maintenance.

The expert shell(VP-Expert), while user friendly, proved to be inefficient in building and supporting the atomic nature of the maintenance operations and the integrity constraints. VP-Expert was not designed to access a database efficiently. The limitation of single record access commands, such as GET and PUT, severely inhibits the performance of the shell in any query operations on medium to large databases.

The validity of using an expert system as a front end to check potential violations of one or more integrity constraints was proved. Naturally, the correctness of all values in the database could not be guaranteed. Any semantic integrity system could only ensure that the data in the database meet the integrity constraints defined in the system.

B. RECOMMENDATIONS

Initially, this researcher attempted to use an expert shell other than VP-Expert to develop the front end system. A Structured Query Language Interface (VP-Expert/SQL) was the first choice. It was hoped that using this system would provide a powerful tool for the enforcement of integrity

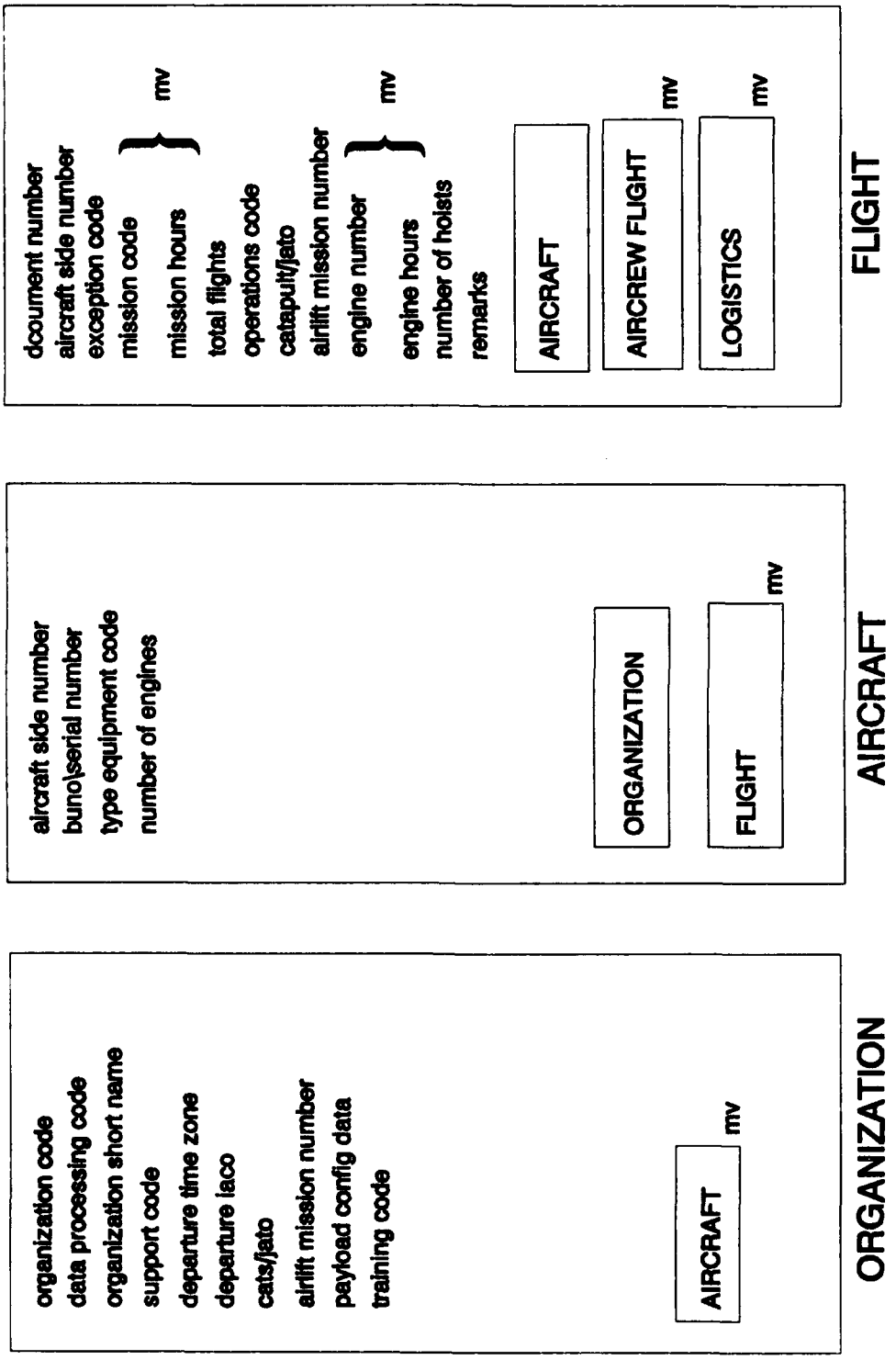
constraints within a relational database. However, this software proved to be unstable and was recently withdrawn, along with all technical support. This was unfortunate, but SQL should still be considered a feasible tool for follow-up research in this area. SQL would enable subqueries and join operations and eliminate many of the inefficiencies inherent to the system (i.e., loops, nested loops. see Appendix E)

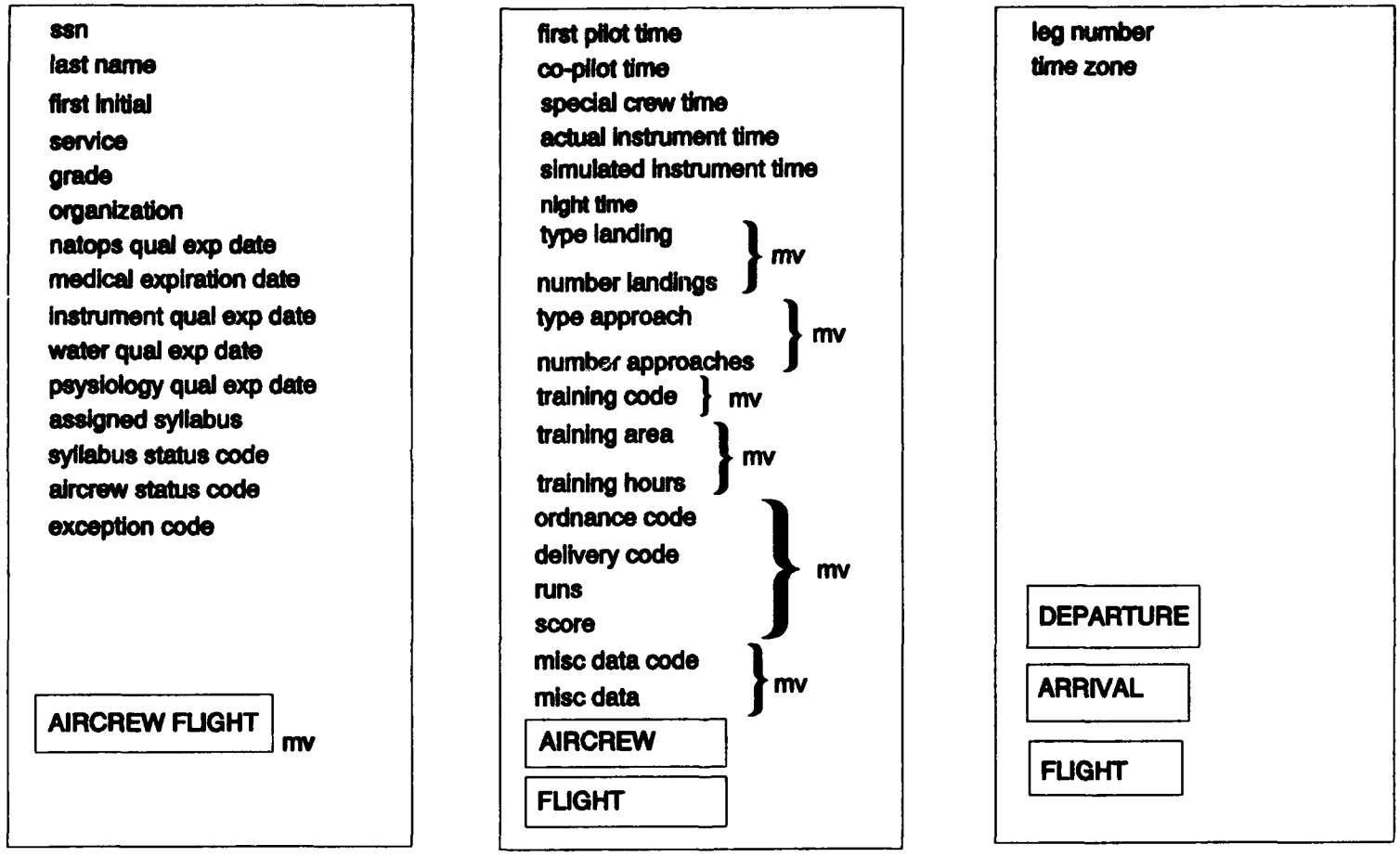
The prototype front end expert system developed in this thesis resulted in a knowledge base of approximately 150 rules. If the number of rules increase, the opportunity for redundant and possibly conflicting rules would multiply. This would inhibit the process of revalidating the system after making changes to the knowledge base. The importance of checking the knowledge base becomes even greater as this happens.

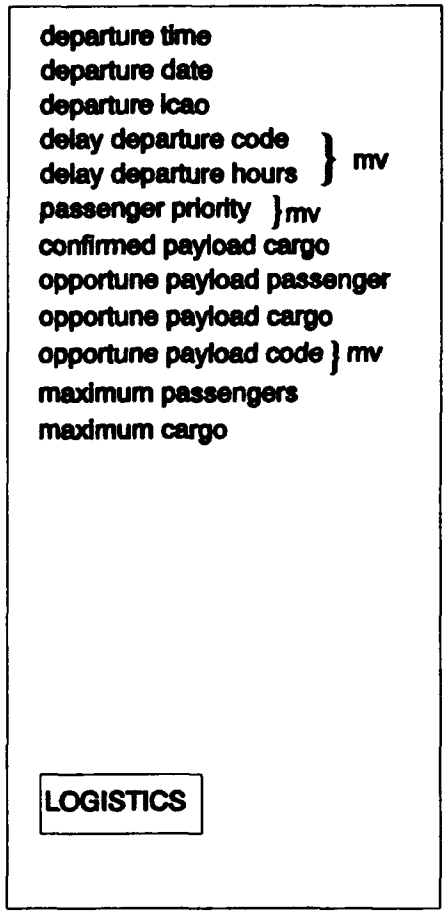
Other follow-up research may include the feasibility of using an object oriented database in providing semantic integrity. Object oriented languages provide for the notion of objects, classes, and inheritance. As opposed to tuples in the relational model, objects have an identity which is independent of their value. This characteristic is central to the domain concept and should enhance this approach to enforce integrity.

APPENDIX A

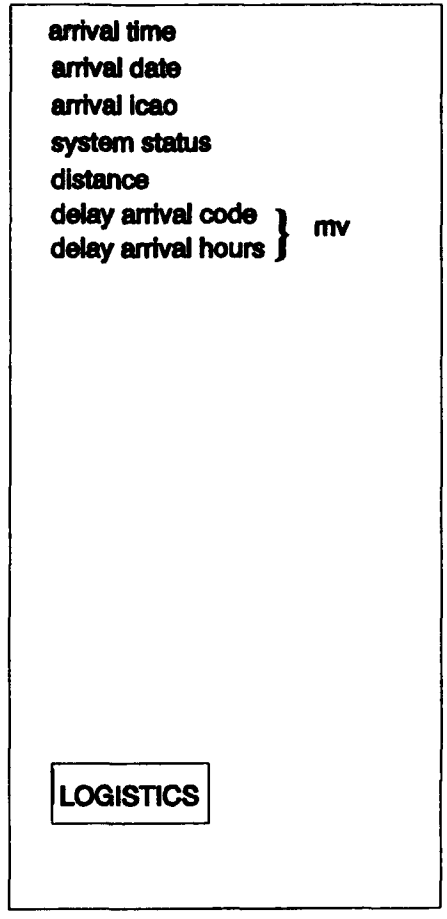
NAVAL AIRCRAFT FLIGHT RECORD OBJECT DIAGRAMS




ssn
last name
first initial
service
grade
organization
natops qual exp date
medical expiration date
instrument qual exp date
water qual exp date
psysiology qual exp date
assigned syllabus
syllabus status code
aircrew status code
exception code
AIRCREW FLIGHT
mv
AIRCREW
first pilot time
co-pilot time
special crew time
actual instrument time
simulated instrument time
night time
type landing
number landings
mv
type approach
number approaches
mv
training code
mv
training area
training hours
mv
ordnance code
delivery code
runs
score
mv
misc data code
misc data
mv
AIRCREW
FLIGHT
AIRCREW FLIGHT
leg number
time zone
DEPARTURE
ARRIVAL
FLIGHT
LOGISTICS



DEPARTURE



ARRIVAL

APPENDIX B

NAVAL AIRCRAFT FLIGHT RECORD OBJECT SPECIFICATIONS

Object Definitions

FLIGHT OBJECT
document number; docnum
aircraft side number; sidenum
exception code; excd
mission code; msn MV
mission hours; hours MV
total flights; totflt
operations code; ops
catapult/jato; cj
airlift mission number; misnum
engine number; engnum MV
engine hours; hours MV
number of hoists; numhoists
remarks; remarks
AIRCRAFT; AIRCRAFT object; SUBSET [aircraft side number]
AIRCREW FLIGHT; AIRCREW FLIGHT object, MV
LOGISTICS; LOGISTICS object; MV

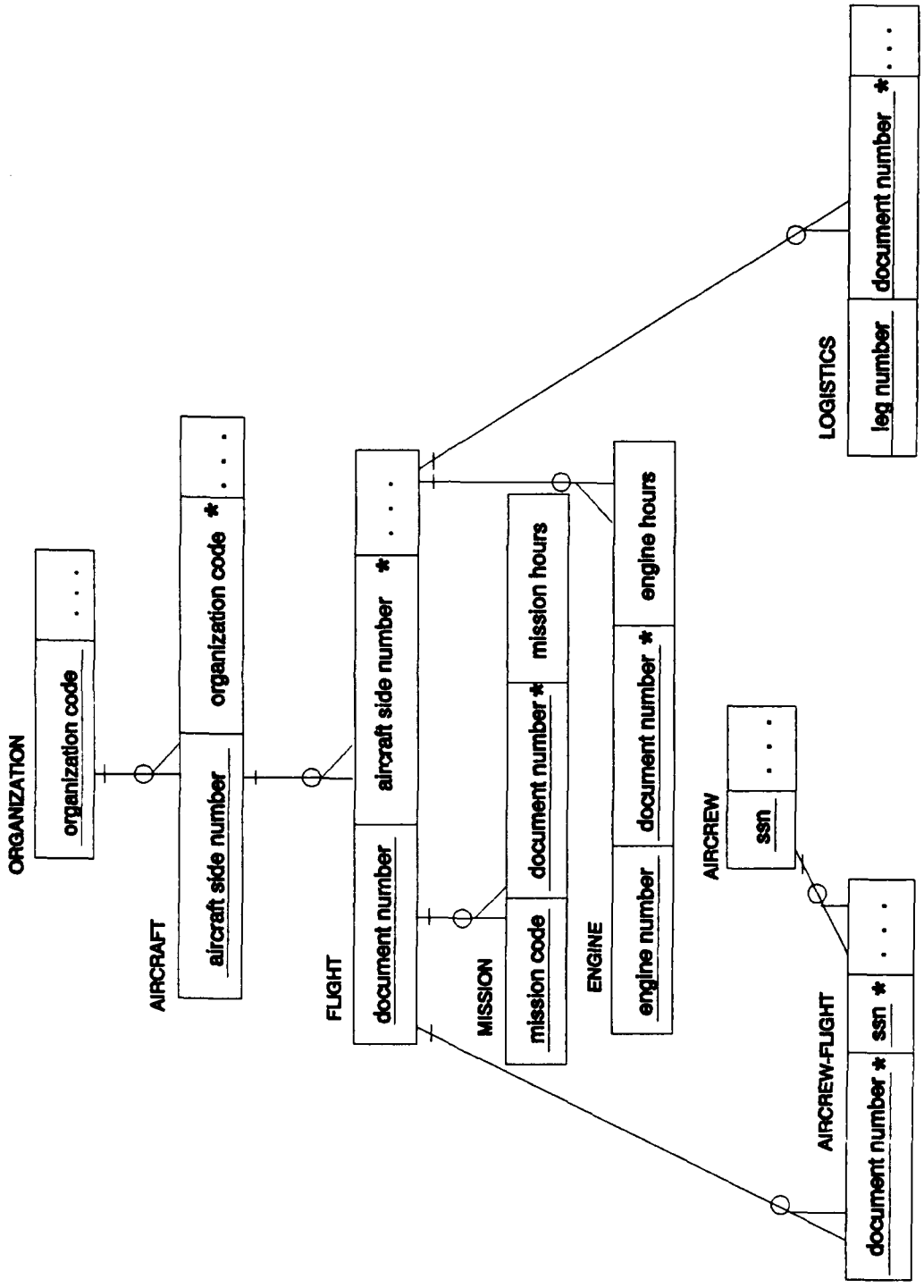
Domain Definitions

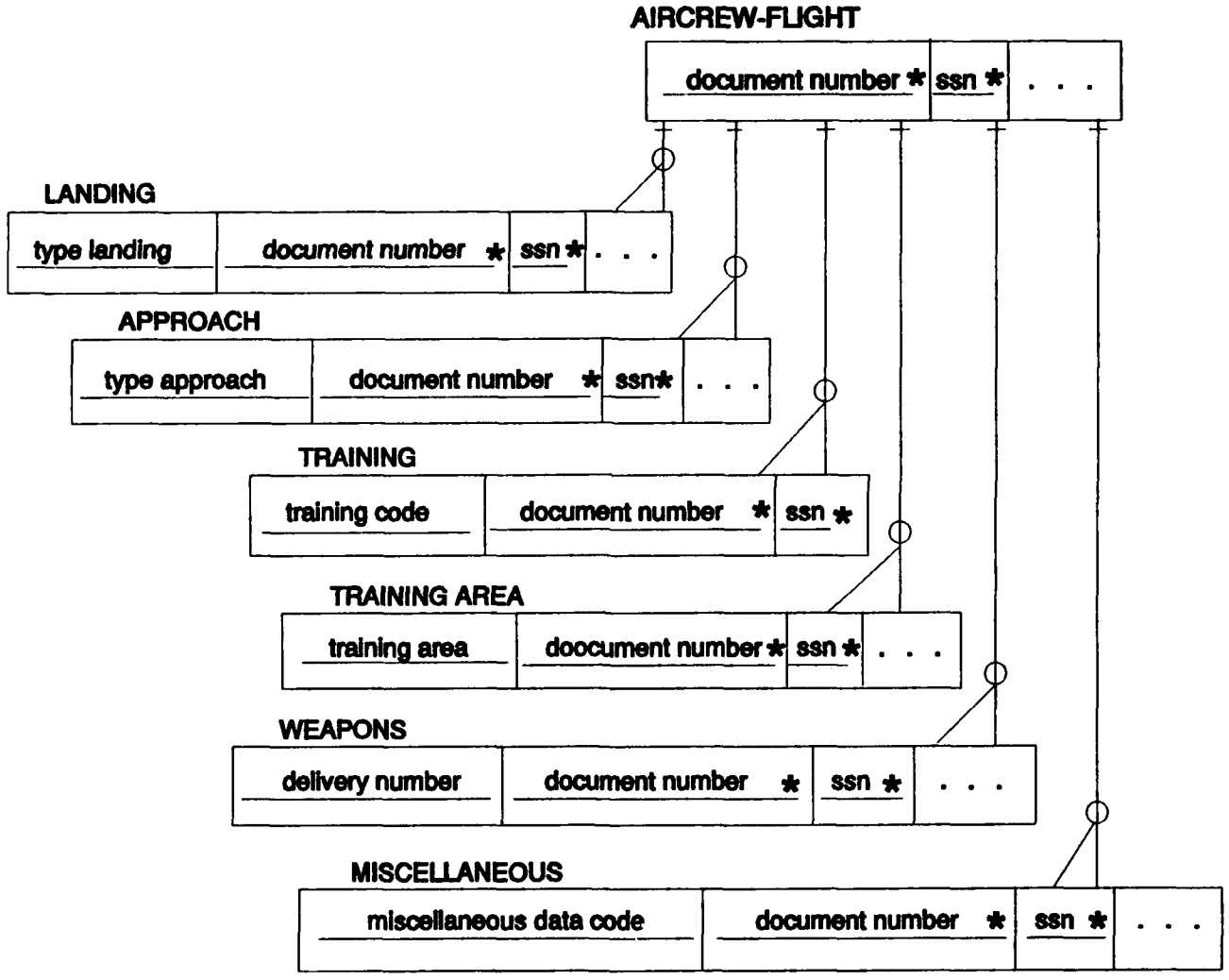
docnum;
    Text 7
    Unique number for organization's Naval Flight Record
sidenum;
    Numeric 5
    Unique number of an organizations aircraft
excd;
    Text 1
    Code to record other than routine flight
msn;
    Text 3, mask FGS,
        where F is the Flight Purpose Code - numeric
              G is the General Purpose Code - alpha
            S is the Specific Purpose Code - numeric
    Unique mission code for a specific flight
hours;
    Numeric 3, mask 99.9
    Hours dedicated to performance of mission
totflt;
    Numeric 2
    Total number of flights
ops;
    Text 1
    Code for ship/shore operational scenario

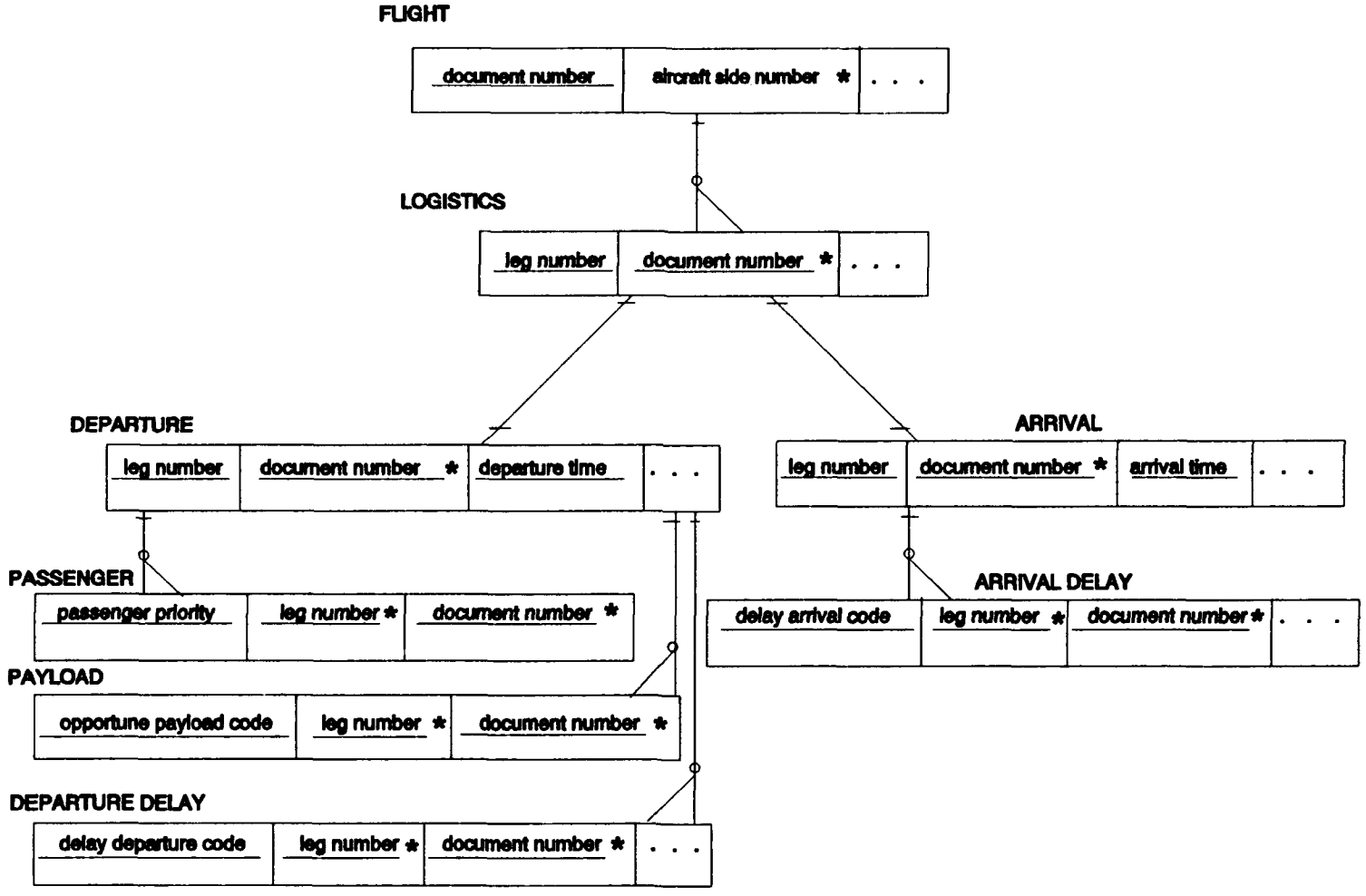
cj;
    Numeric 1
    Total number of catapult/jet assisted takeoff launches
misnum;
    Text 9, mask ORGDATENN,
        Where ORG is the organization code - Text
            DATE is the julian date - numeric
            NN is 01-99 sequentially assigned
engnum;
    Numeric 1
    Unique engine number
numhoists;
    Numeric 2
    Total number of hoists on a flight
remarks;
    Text 15
    Used as needed

APPENDIX C

NAVAL AIRCRAFT FLIGHT RECORD RELATIONAL DIAGRAMS

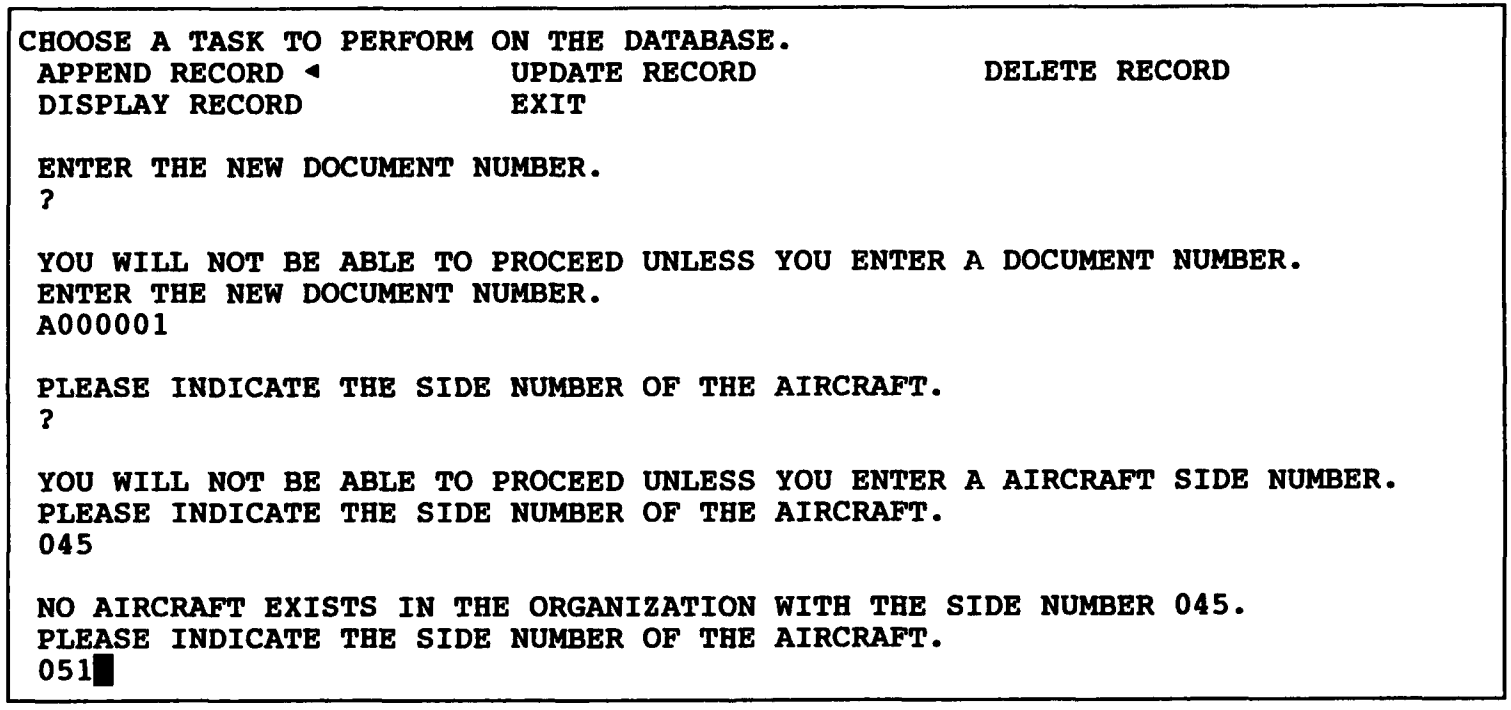


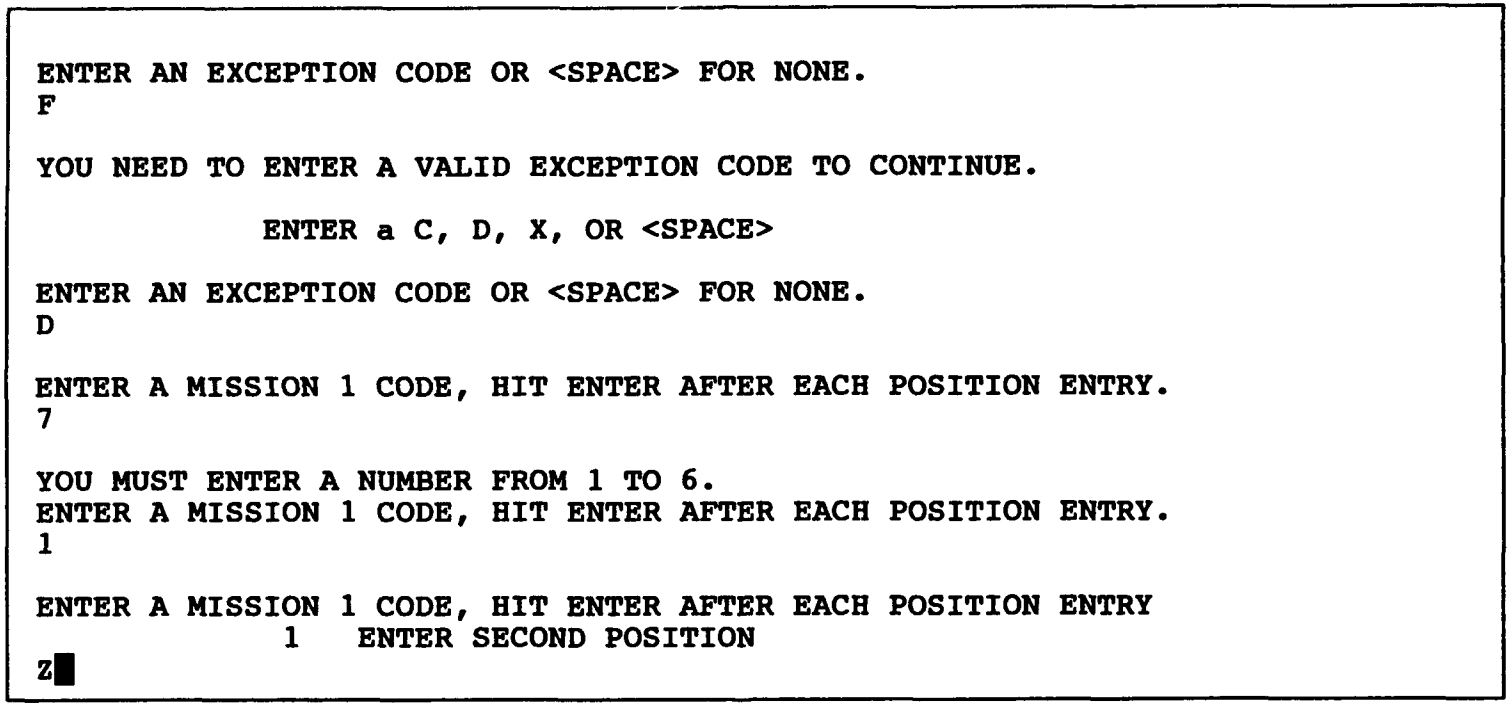

AIRCREW-FLIGHT
document number *
ssn *
. . .
LANDING
type landing
document number *
ssn *
. . .
APPROACH
type approach
document number *
ssn *
. . .
TRAINING
training code
document number *
ssn *
TRAINING AREA
training area
doocument number *
ssn *
. . .
WEAPONS
delivery number
document number *
ssn *
. . .
MISCELLANEOUS
miscellaneous data code
document number *
ssn *
. . .


FLIGHT
document number
aircraft side number *
. . .
LOGISTICS
leg number
document number *
. . .
DEPARTURE
leg number
document number *
departure time
. . .
ARRIVAL
leg number
document number *
arrival time
. . .
PASSENGER
passenger priority
leg number *
document number *
ARRIVAL DELAY
delay arrival code
leg number *
document number *
. . .
PAYLOAD
opportune payload code
leg number *
document number *
DEPARTURE DELAY
delay departure code
leg number *
document number *
. . .

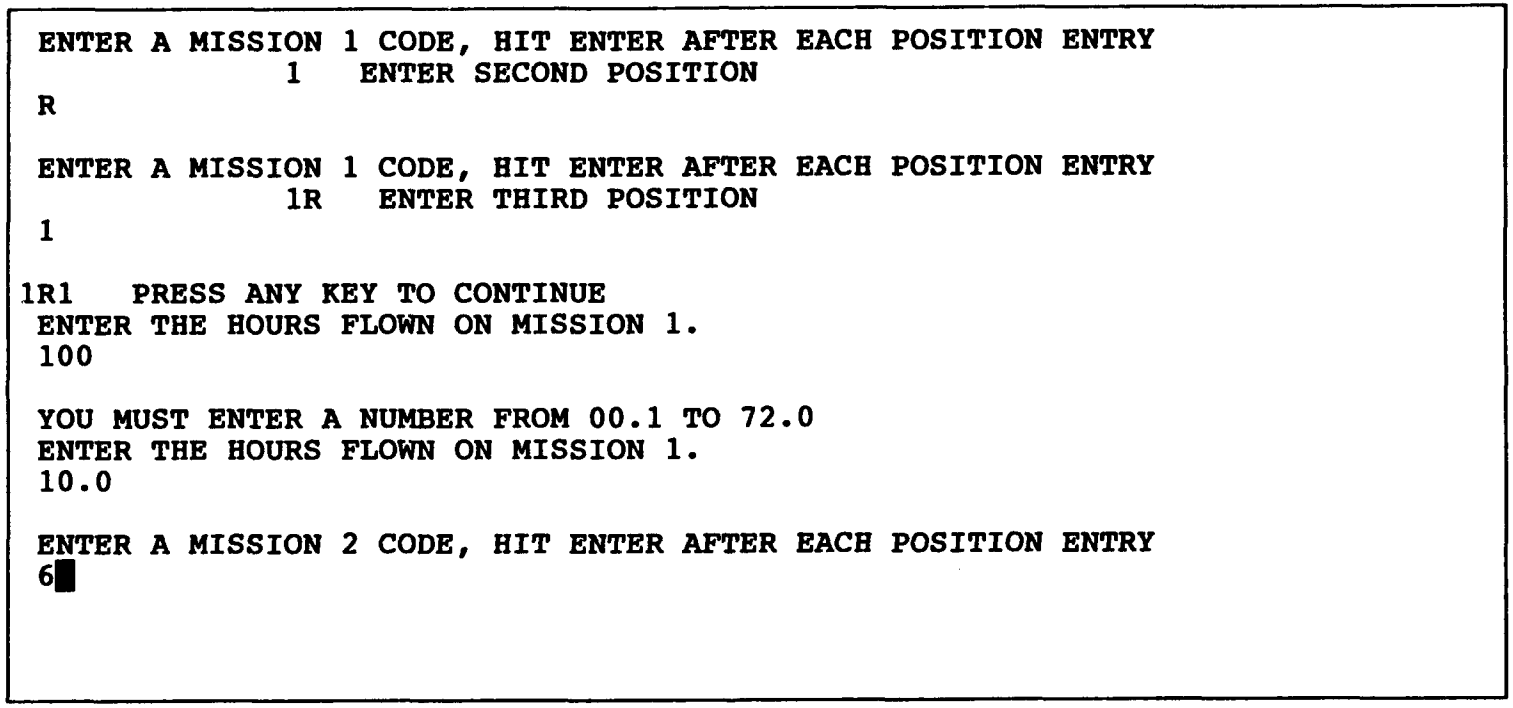
APPENDIX D
SESSION WITH NAVAL AIRCRAFT FLIGHT RECORD EXPERT SYSTEM

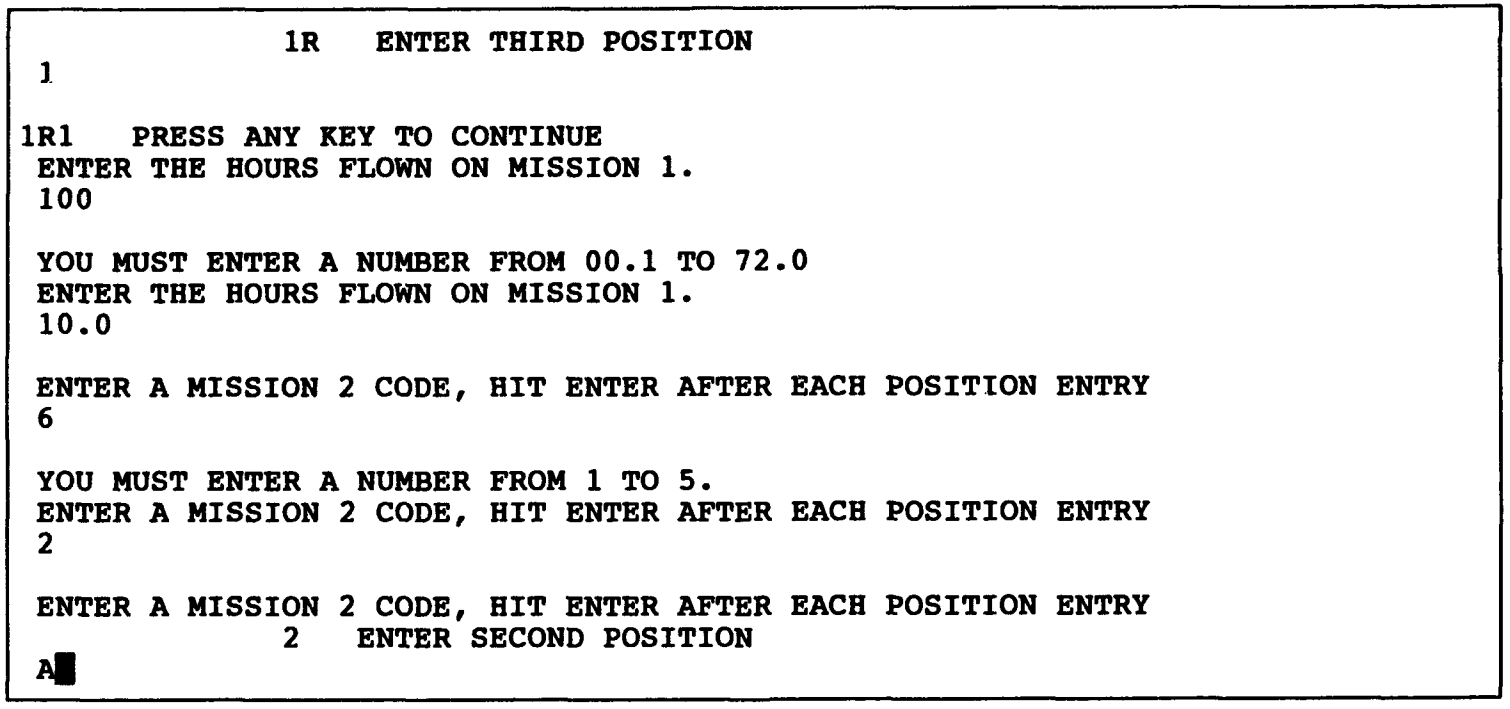
```
THIS IS A FRONT END INTEGRITY EXPERT SYSTEM TO ENABLE
THE ACCURATE COLLECTION OF INFORMATION FOR THE NAVYS
AIRCRAFT FLIGHT RECORD, OPNAV 3710/4.

PRESS ANY KEY TO BEGIN...

1Help  2Go  3WhatIf  4Variable 5Rule  6Set  7Edit  8Quit
1Help 2How? 3Why? 4Slow 5Fast 6Quit
```

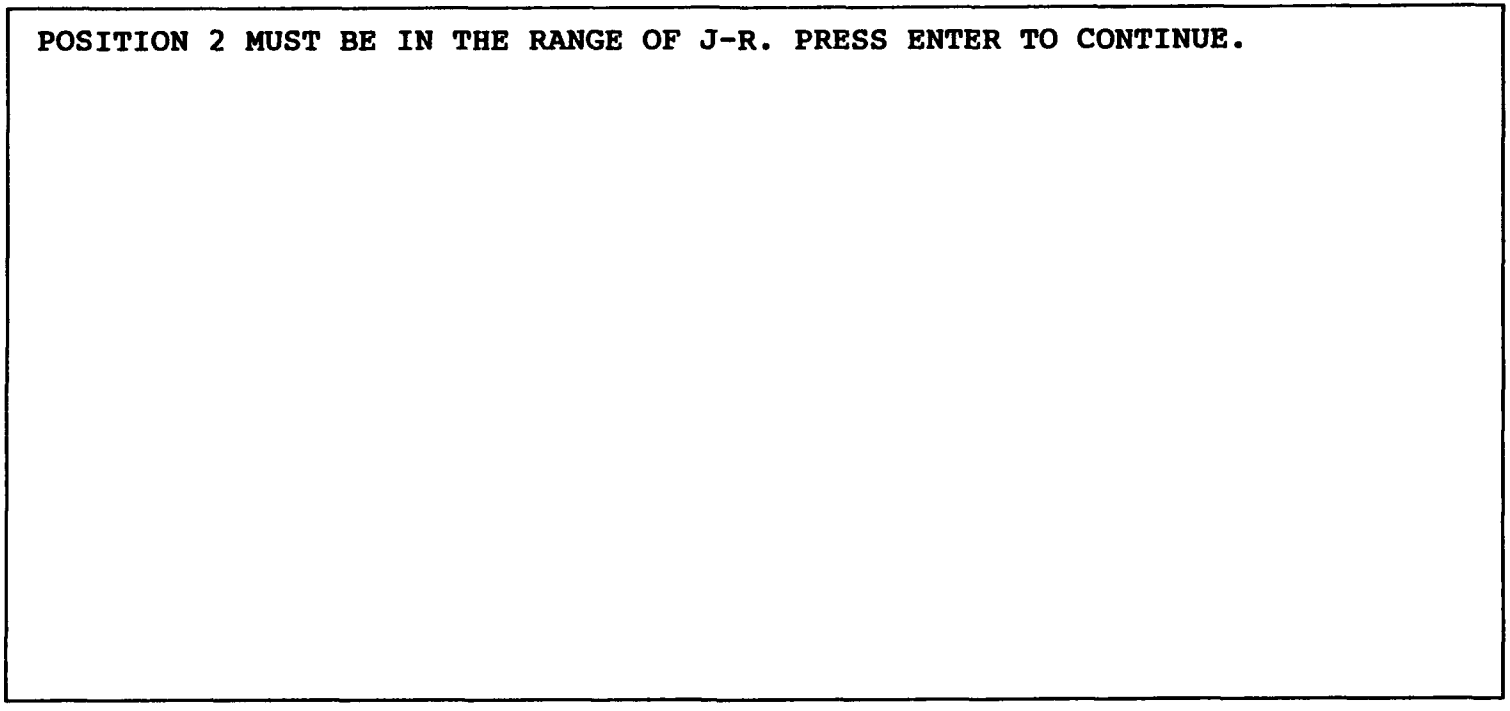

CHOOSE A TASK TO PERFORM ON THE DATABASE.
APPEND RECORD ◄
UPDATE RECORD
DELETE RECORD
DISPLAY RECORD
EXIT
ENTER THE NEW DOCUMENT NUMBER.
?
YOU WILL NOT BE ABLE TO PROCEED UNLESS YOU ENTER A DOCUMENT NUMBER.
ENTER THE NEW DOCUMENT NUMBER.
A000001
PLEASE INDICATE THE SIDE NUMBER OF THE AIRCRAFT.
?
YOU WILL NOT BE ABLE TO PROCEED UNLESS YOU ENTER A AIRCRAFT SIDE NUMBER.
PLEASE INDICATE THE SIDE NUMBER OF THE AIRCRAFT.
045
NO AIRCRAFT EXISTS IN THE ORGANIZATION WITH THE SIDE NUMBER 045.
PLEASE INDICATE THE SIDE NUMBER OF THE AIRCRAFT.
051
Enter to select ? & Enter for Unknown /Q to quit

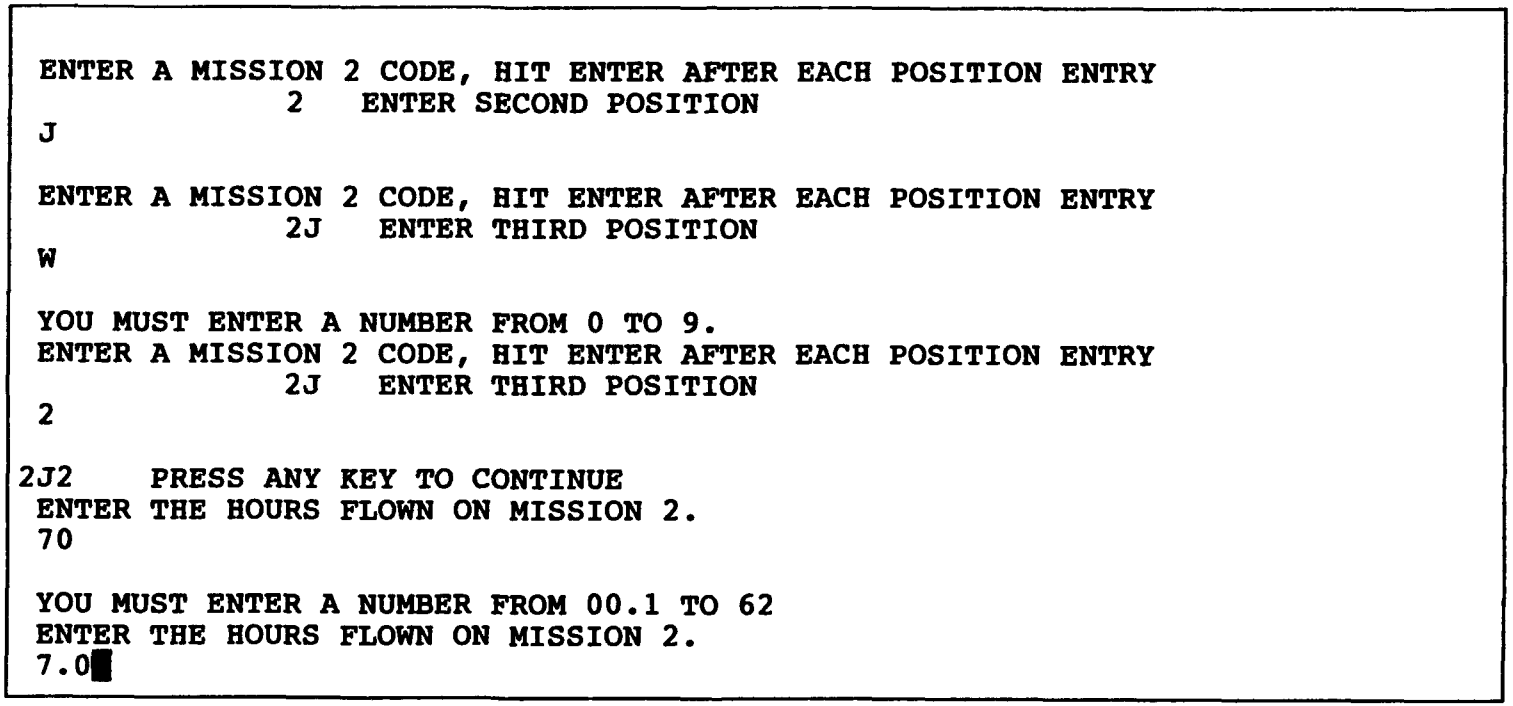

ENTER AN EXCEPTION CODE OR <SPACE> FOR NONE.
F
YOU NEED TO ENTER A VALID EXCEPTION CODE TO CONTINUE.
ENTER a C, D, X, OR <SPACE>
ENTER AN EXCEPTION CODE OR <SPACE> FOR NONE.
D
ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH POSITION ENTRY.
7
YOU MUST ENTER A NUMBER FROM 1 TO 6.
ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH POSITION ENTRY.
1
ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH POSITION ENTRY
1 ENTER SECOND POSITION
Z
Enter to select ? & Enter for Unknown /Q to quit

POSITION 2 MUST BE R, A-I, OR N-P. PRESS ENTER TO CONTINUE.

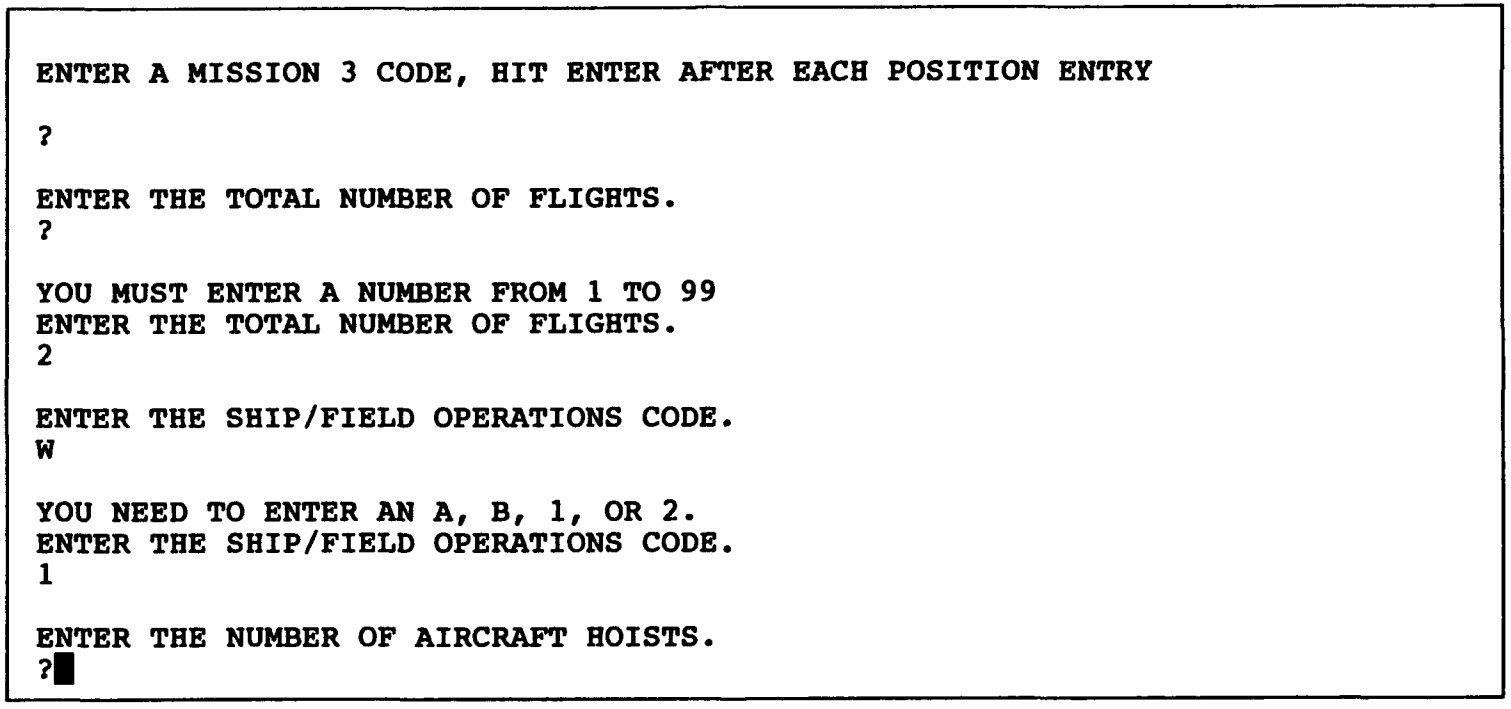

ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH POSITION ENTRY
1 ENTER SECOND POSITION
R
ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH POSITION ENTRY
1R ENTER THIRD POSITION
1
1R1 PRESS ANY KEY TO CONTINUE
ENTER THE HOURS FLOWN ON MISSION 1.
100
YOU MUST ENTER A NUMBER FROM 00.1 TO 72.0
ENTER THE HOURS FLOWN ON MISSION 1.
10.0
ENTER A MISSION 2 CODE, HIT ENTER AFTER EACH POSITION ENTRY
6
Enter to select ? & Enter for Unknown /Q to quit


1R ENTER THIRD POSITION
1
1R1 PRESS ANY KEY TO CONTINUE
ENTER THE HOURS FLOWN ON MISSION 1.
100
YOU MUST ENTER A NUMBER FROM 00.1 TO 72.0
ENTER THE HOURS FLOWN ON MISSION 1.
10.0
ENTER A MISSION 2 CODE, HIT ENTER AFTER EACH POSITION ENTRY
6
YOU MUST ENTER A NUMBER FROM 1 TO 5.
ENTER A MISSION 2 CODE, HIT ENTER AFTER EACH POSITION ENTRY
2
ENTER A MISSION 2 CODE, HIT ENTER AFTER EACH POSITION ENTRY
2 ENTER SECOND POSITION
A
Enter to select ? & Enter for Unknown /Q to quit

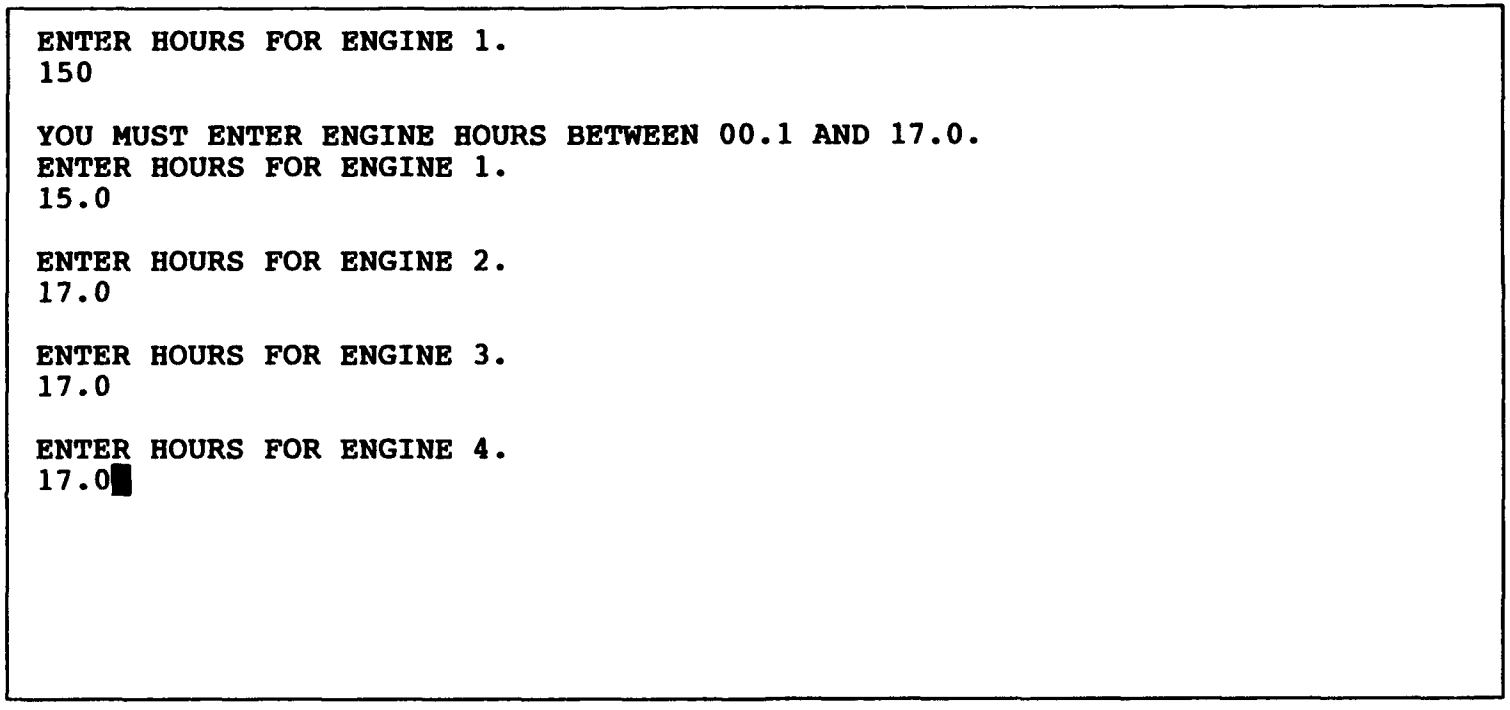

POSITION 2 MUST BE IN THE RANGE OF J-R. PRESS ENTER TO CONTINUE.



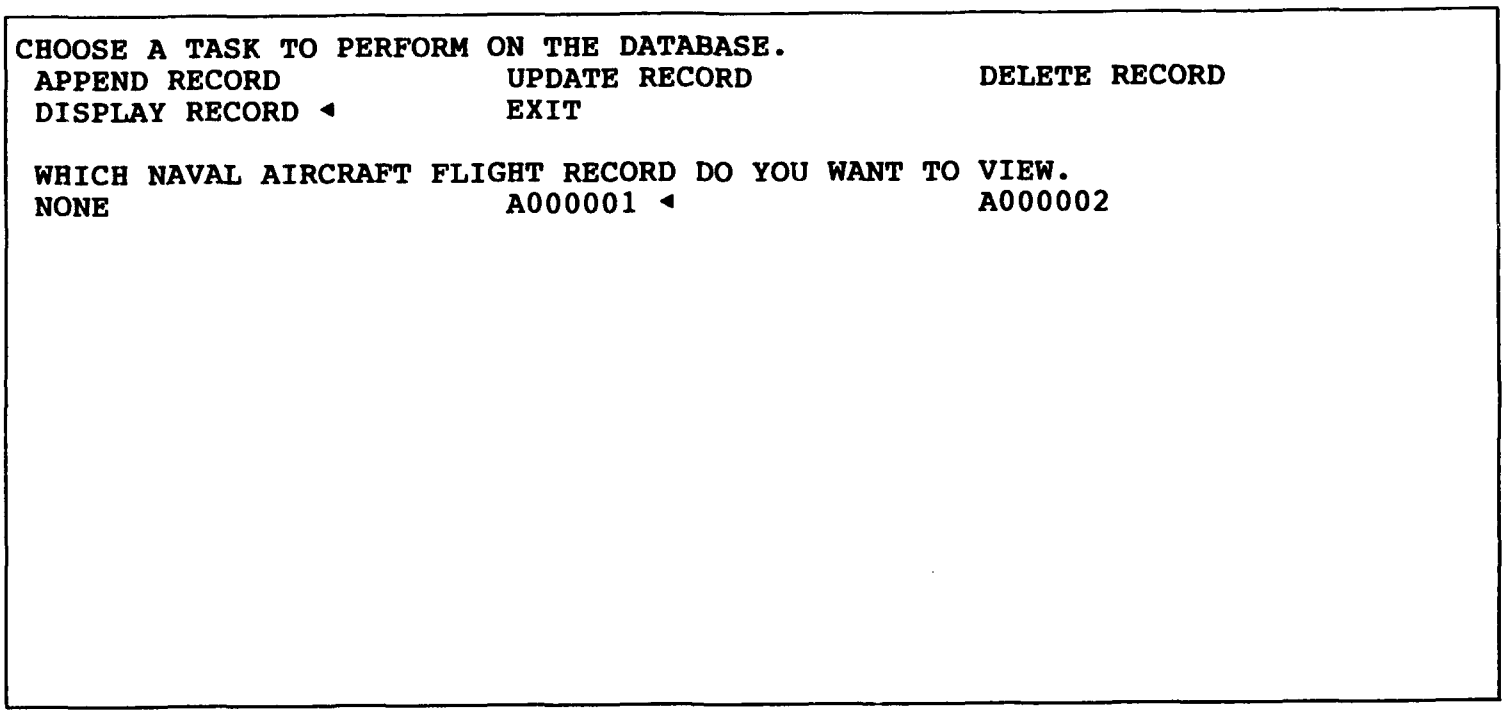
Enter to select ? & Enter for Unknown /Q to quit

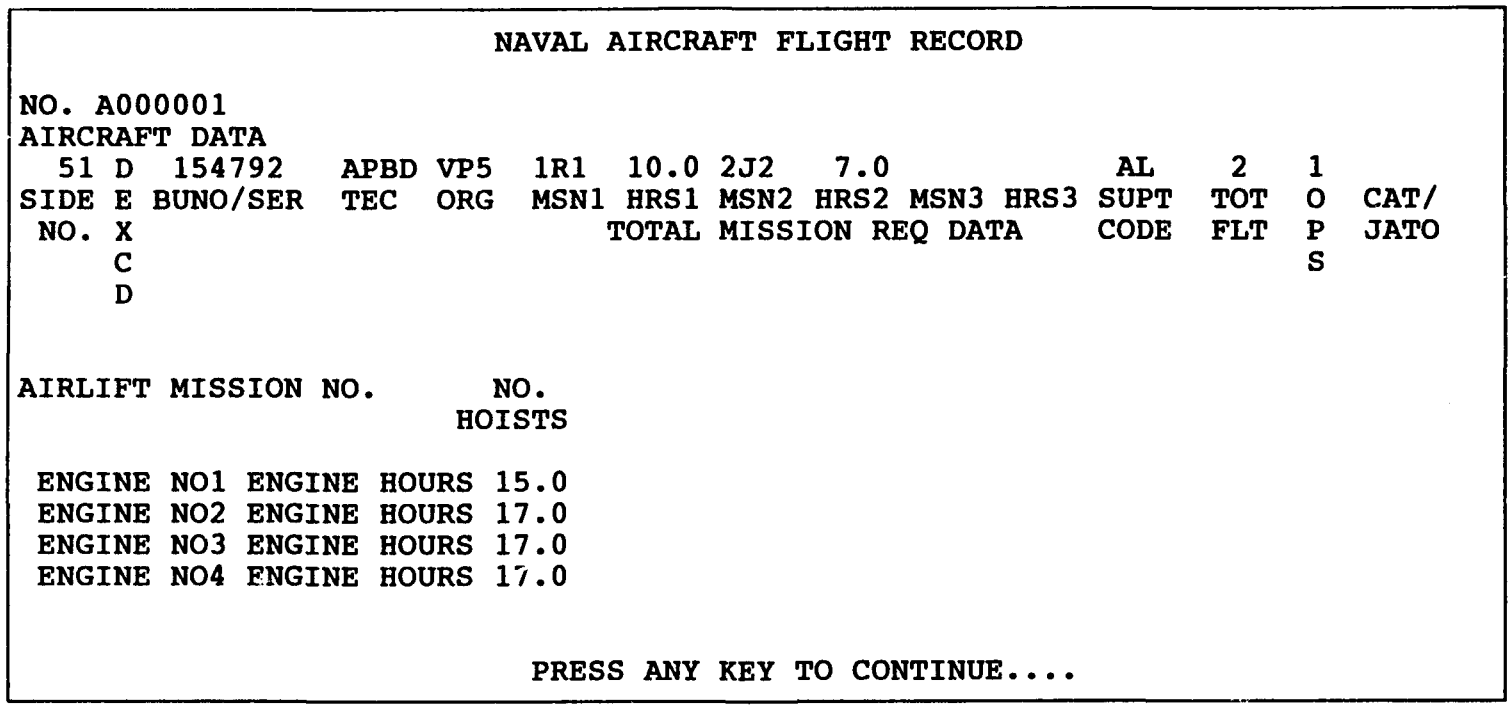


Enter to select ? & Enter for Unknown /Q to quit



Enter to select ? & Enter for Unknown /Q to quit


CHOOSE A TASK TO PERFORM ON THE DATABASE.
APPEND RECORD
UPDATE RECORD
DELETE RECORD
DISPLAY RECORD ◄
EXIT
WHICH NAVAL AIRCRAFT FLIGHT RECORD DO YOU WANT TO VIEW.
NONE
A000001 ◄
A000002



NAVAL AIRCRAFT FLIGHT RECORD

NO. A000001
AIRCRAFT DATA

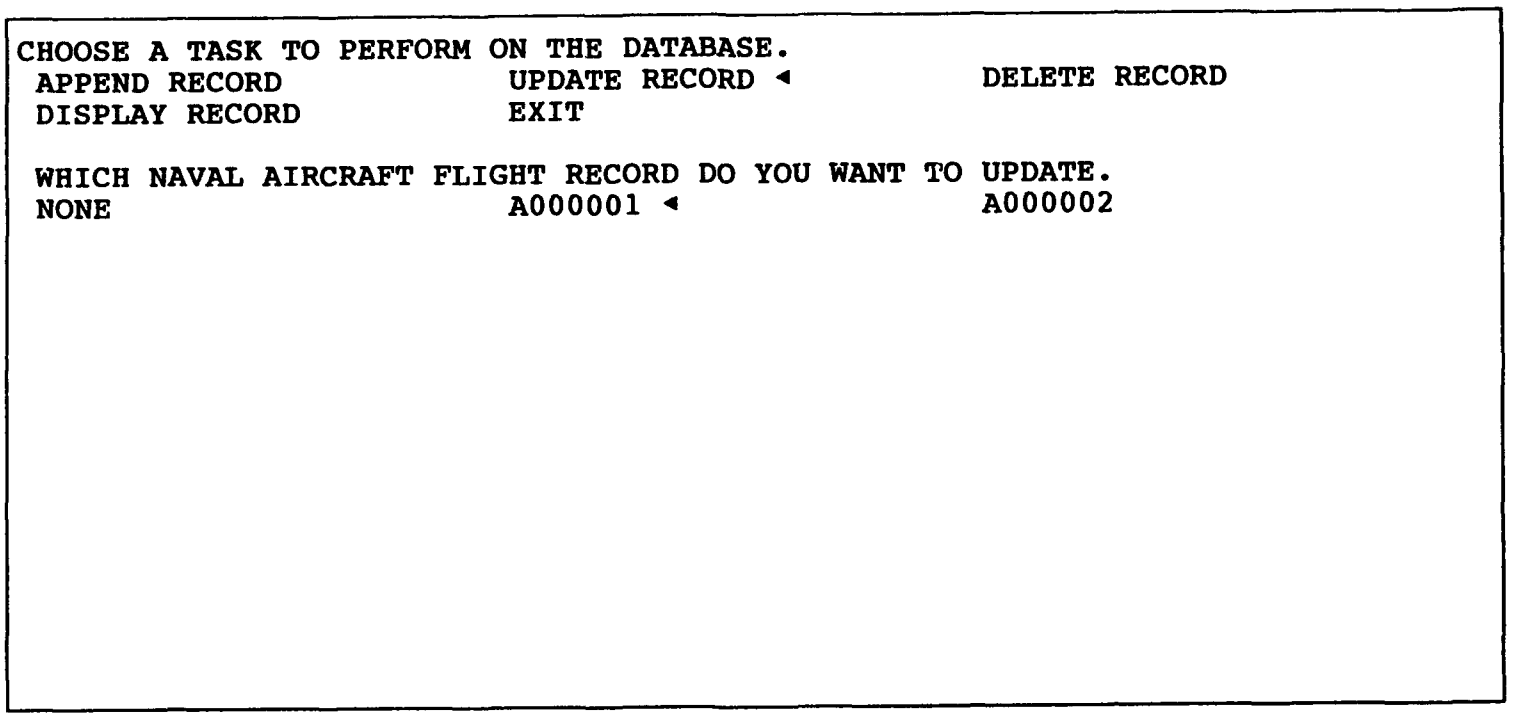
SIDE NO.	DEXCD	BUNO/SER	TEC	ORG	MSN1	HRS1	MSN2	HRS2	MSN3	HRS3	SUPT CODE	TOT FLT	OPS	CAT/ JATO
51	D	154792	APBD	VP5	1R1	10.0	2J2	7.0			AL	2	1	
					TOTAL MISSION REQ DATA									

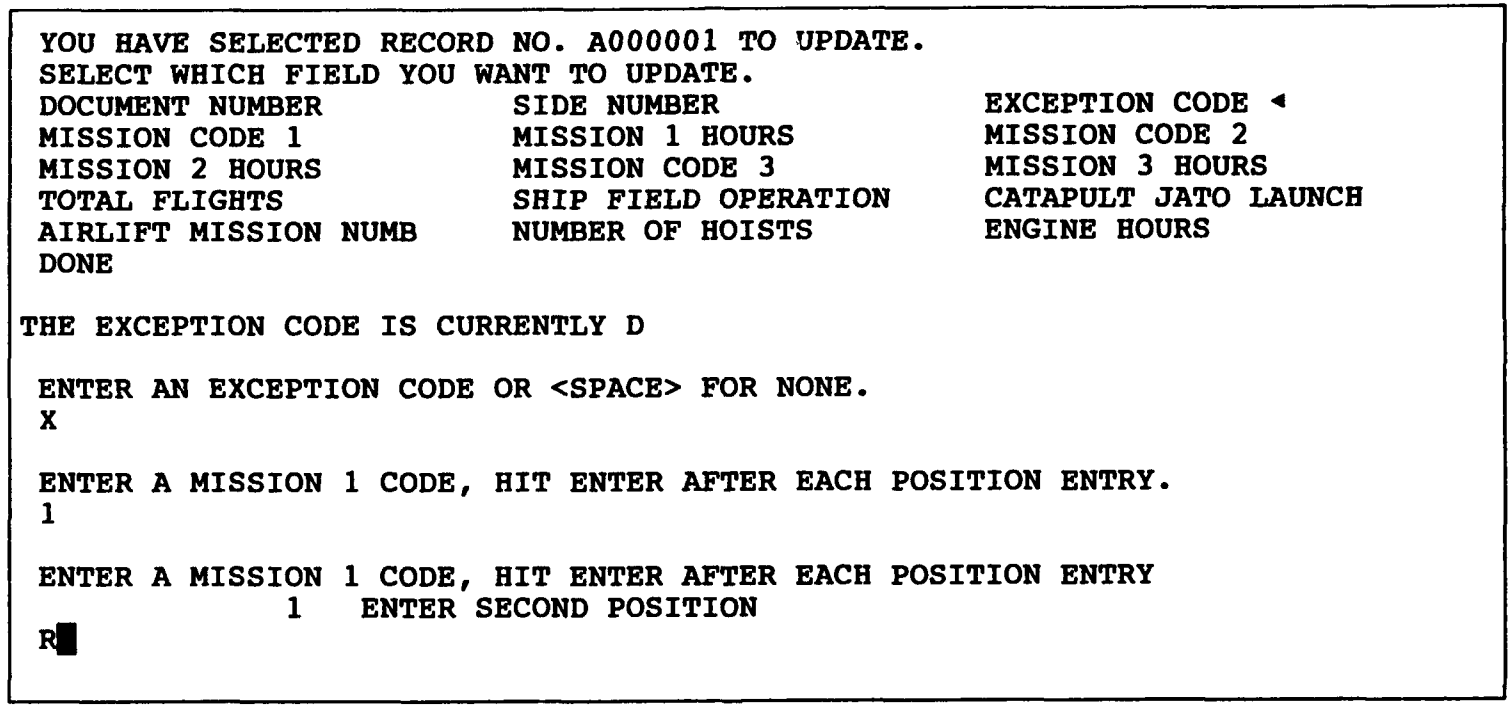
AIRLIFT MISSION NO.

NO. HOISTS

ENGINE NO1 ENGINE HOURS 15.0
ENGINE NO2 ENGINE HOURS 17.0
ENGINE NO3 ENGINE HOURS 17.0
ENGINE NO4 ENGINE HOURS 17.0

PRESS ANY KEY TO CONTINUE....

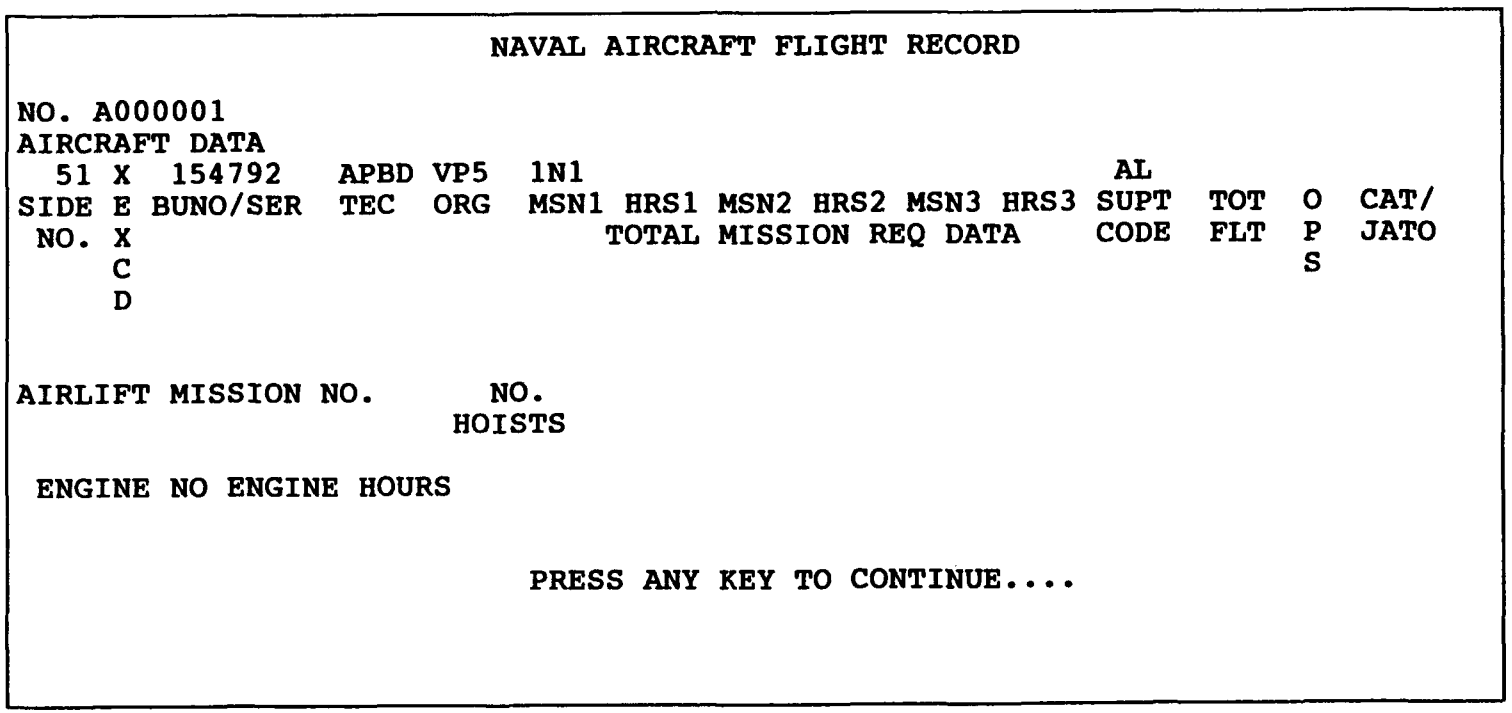

CHOOSE A TASK TO PERFORM ON THE DATABASE.
APPEND RECORD
UPDATE RECORD ◄
DELETE RECORD
DISPLAY RECORD
EXIT
WHICH NAVAL AIRCRAFT FLIGHT RECORD DO YOU WANT TO UPDATE.
NONE
A000001 ◄
A000002

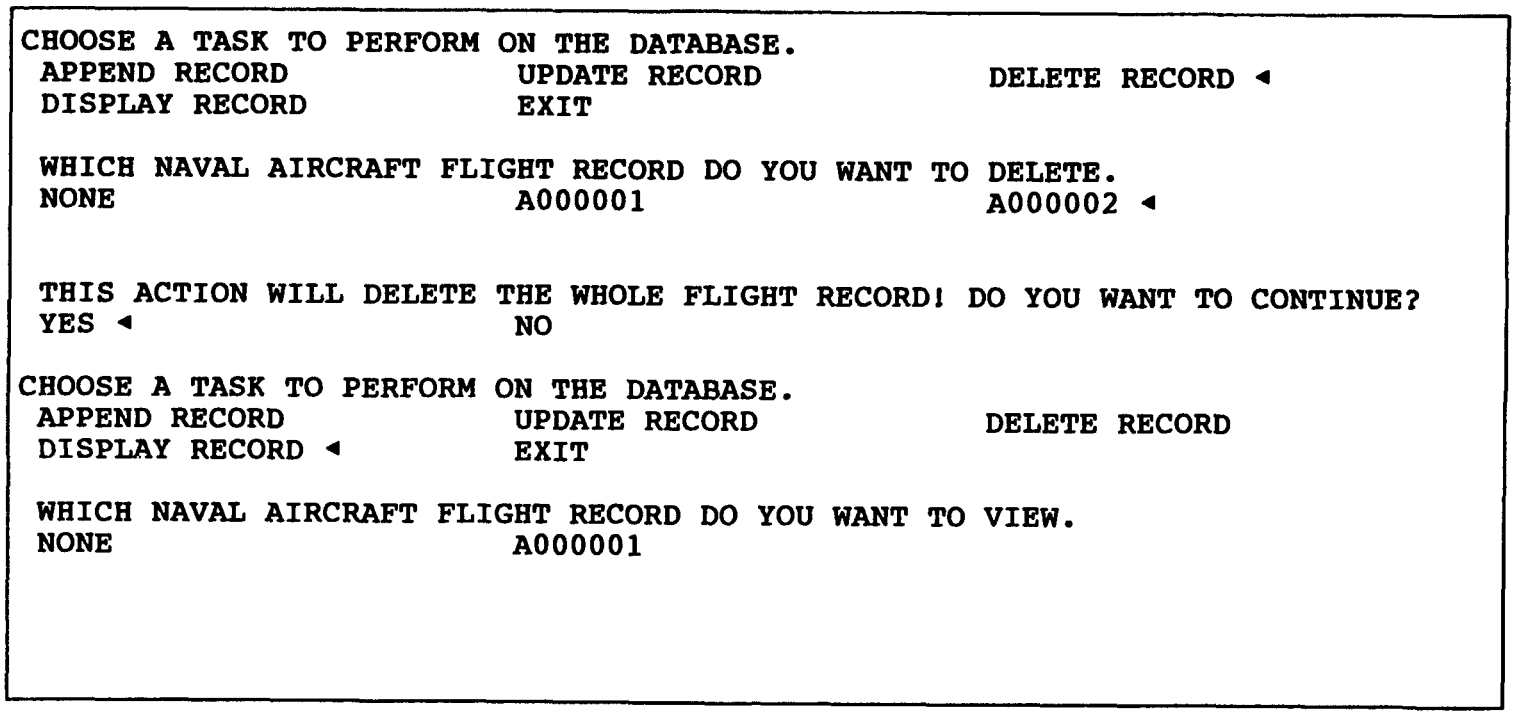


Enter to select ? & Enter for Unknown /Q to quit

POSITION 2 MUST BE N, OR O. PRESS ENTER TO CONTINUE.

```
ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH POSITION ENTRY
              1   ENTER SECOND POSITION
N

ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH POSITION ENTRY
              1N   ENTER THIRD POSITION
1

1N1   PRESS ANY KEY TO CONTINUE
```


NAVAL AIRCRAFT FLIGHT RECORD
NO. A000001
AIRCRAFT DATA
51 X 154792 APBD VP5 1N1
SIDE E BUNO/SER TEC ORG MSN1 HRS1 MSN2 HRS2 MSN3 HRS3 AL SUPT TOT O CAT/
NO. X TOTAL MISSION REQ DATA CODE FLT P JATO
C S
D
AIRLIFT MISSION NO. NO. HOISTS
ENGINE NO ENGINE HOURS
PRESS ANY KEY TO CONTINUE....



↑ ↓ → Enter to select END to complete /Q to Quit ? for Unknown

APPENDIX E

NAVAL AIRCRAFT FLIGHT RECORD RULE-BASE

```
! Naval Aircraft Flight Record Expert System
! By George J. Salitsky
! Naval Postgraduate School
! This program is a prototype Front End Expert System
! designed to maintain semantic integrity within the
! database according to the integrity constraints specified
! in the knowledge base.


AUTOQUERY;
RUNTIME;
ENDOFF;

ACTIONS
FORMAT TOTAL,  4.1
DISPLAY "THIS IS A FRONT END INTEGRITY EXPERT SYSTEM TO
      ENABLE THE ACCURATE COLLECTION OF INFORMATION FOR THE
            NAVYS AIRCRAFT FLIGHT RECORD, OPNAV 3710/4.



                              PRESS ANY KEY TO BEGIN...~"

! loop to ask user which maintenance operation to perform on
! the database. whichtask is the main menu, options include:
!          1. APPEND
!          2. UPDATE
!          3. DELETE
!          4. DISPLAY
!          5. EXIT

CLS
RESET WHICHTASK
     WHILETRUE WHICHTASK <> EXIT THEN
          RESET ALL
! set up variable BLANK
          CHR 32, BLANK
          FIND WHICHTASK
          FIND TASKCOMPLETED
     END;

! ********************* APPEND OPERATION ********************

RULE APPEND_RECORD
IF
```

```
     WHICHTASK = APPEND_RECORD
THEN
     FIELD_TO_UPDATE = NONE
     TASKCOMPLETED = YES
     RESET DOCNUM_NEW
! ask user for document number
     FIND DOCNUM_NEW
     RESET DOCNUM_NOT_MISSING
! cannot allow a null value for document number
     FIND DOCNUM_NOT_MISSING
     RESET DOCNUM_DUPLICATE
! cannot allow duplicate document numbers
     FIND DOCNUM_DUPLICATE
     DOCNUM = (DOCNUM_NEW)
     CLOSE FLIGHT
     RESET SIDENUM_NEW
! ask user for aircraft side number
     FIND SIDENUM_NEW
     RESET SIDENUM_NOT_MISSING
! cannot allow a null value for side number
     FIND SIDENUM_NOT_MISSING
     RESET SIDENUM_EXISTS
! side number must match an aircraft in organization
     FIND SIDENUM_EXISTS
     CLOSE AIRCRAFT
     RESET EXCD
! ask user for exception code
     FIND EXCD
     RESET EXCD_VALID
! only certain exception codes allowed
     FIND EXCD_VALID
! find mission code 1 position 1
     CLS
     RESET MSN1_1
     FIND MSN1_1
     RESET MSN11_VALID
     FIND MSN11_VALID
! find mission code 1 position 2
     CLS
     RESET MSN1_2
     FIND MSN1_2
     RESET MSN12_VALID
     FIND  MSN12_VALID
! find mission code 1 position 3
     CLS
     RESET MSN1_3
     FIND MSN1_3
     RESET MSN13_VALID
     FIND MSN13_VALID
     CLS
! find mission 1 hours
```

```
      RESET CHECK
      FIND CHECK
      TEMPHRS1 = 0
      TEMPHRS2 = 0
      TEMPHRS3 = 0
      TOTHRS = 72.0
      SUBTOTAL = 0
      RESET HRS1_VALID
      FIND HRS1_VALID
! find mission code 2 position 1
      CLS
      RESET MSN21_VALID
      FIND MSN21_VALID
! find mission code 2 position 2
      CLS
      RESET MSN22_VALID
      FIND  MSN22_VALID
! find mission code 2 position 3
      CLS
      RESET MSN23_VALID
      FIND MSN23_VALID
      CLS
! find mission 2 hours
      RESET HRS2_VALID
      FIND HRS2_VALID
! find mission code 3 position 1
      CLS
      RESET MSN31_VALID
      FIND MSN31_VALID
! find mission code 3 position 2
      CLS
      RESET MSN32_VALID
      FIND  MSN32_VALID
! find mission code 3 position 3
      CLS
      RESET MSN33_VALID
      FIND MSN33_VALID
      CLS
! find mission 3 hours
      RESET HRS3_VALID
      FIND HRS3_VALID
      CLS
! find total flights
      RESET TOTFLT_VALID
      FIND TOTFLT_VALID
      CLS
! find ship/field operations code
      RESET OPS_VALID
      FIND OPS_VALID
      CLS
! find catapult/jato launches as necessary
```

```
      GET ALL, ORGAN, CATSJATO
      RESET CJ_VALID
      FIND CJ_VALID
      CLOSE ORGAN
      CLS
! find airlift mission number as necessary
      GET ALL, ORGAN, AIRLIFT
      RESET AIRLIFT_VALID
      FIND AIRLIFT_VALID
      CLOSE ORGAN
      CLS
! find number of hoists
      RESET NUMHOIST_VALID
      FIND NUMHOIST_VALID
      CLS
! append new record to flight database
      APPEND FLIGHT
! loop to get engine hours for aircraft on flight
! determined by aircraft record
      GET SIDENUM = (SIDENUM_NEW), AIRCRAFT, ENGINES
      CLOSE AIRCRAFT
      RESET ENGHRS_VALID
      FIND ENGHRS_VALID
      CLS;

! ********************* UPDATE OPERATION ********************

RULE UPDATE_DOCUMENT
IF
      WHICHTASK = UPDATE_RECORD
THEN
      TASKCOMPLETED = YES
      RESET DOCNUM_UPDATE
      MENU DOCNUM_UPDATE, ALL, FLIGHT, DOCNUM
! ask user for document number from menu of all document
! numbers
      FIND DOCNUM_UPDATE
      MRESET DOCNUM_UPDATE
      RESET UPDATE
      FIND UPDATE;

! determine if there are any Flight Records to update
RULE UPDATE
IF
      DOCNUM_UPDATE = NONE AND
      UPDATE = UNKNOWN
THEN
! no flight records to update
      UPDATE = NO
      DISPLAY "       THERE IS NO FLIGHT RECORD TO UPDATE.
```

```
                              PRESS ANY KEY TO CONTINUE
                         ~"
     CLS
ELSE
! flight records available to update
     UPDATE = YES
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     CLOSE FLIGHT
     CLS
     DISPLAY " YOU HAVE SELECTED RECORD NO. {DOCNUM_UPDATE}
TO UPDATE."       RESET FIELD_TO_UPDATE
! ask user for attribute to update by menu field_to_update
     WHILETRUE FIELD_TO_UPDATE <> DONE THEN
          RESET FIELD_TO_UPDATE
          RESET UPDATE_COMPLETED
          FIND FIELD_TO_UPDATE
          FIND UPDATE_COMPLETED
     END;

! ********** UPDATE DOCUMENT NUMBER ********************

RULE UPDATE_DOCUMENT_NUMBER
IF
     FIELD_TO_UPDATE = DOCUMENT_NUMBER
THEN
     UPDATE_COMPLETED = YES
! display current document number
     DISPLAY "{DOCNUM_UPDATE} IS CURRENTLY THE DOCUMENT
               NUMBER.
             "
     CLOSE FLIGHT
     RESET DOCNUM_NEW
! ask user for document number
     FIND DOCNUM_NEW
     RESET DOCNUM_NOT_MISSING
! cannot allow a null value for document number
     FIND DOCNUM_NOT_MISSING
     RESET DOCNUM_DUPLICATE
! cannot allow duplicate document numbers
     FIND DOCNUM_DUPLICATE
     RESET DOCNUM_NOT_MISSING
     RESET DOCNUM_DUPLICATE
     CLOSE FLIGHT
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, DOCNUM
     DOCNUM = (DOCNUM_NEW)
     PUT FLIGHT
     CLOSE FLIGHT
! change document number on ENGINE records
     GET DOCNUM_UPDATE = DOCNUM, FLTENG, DOCNUM
     WHILETRUE DOCNUM <> UNKNOWN THEN
          DOCNUM = (DOCNUM_NEW)
```

```
          PUT FLTENG
          GET DOCNUM_UPDATE = DOCNUM, FLTENG, DOCNUM
     END
     CLOSE FLTENG
     FIELD_TO_UPDATE = DONE;

! *************** UPDATE AIRCRAFT SIDE NUMBER **************

RULE UPDATE_SIDE_NUMBER
IF
     FIELD_TO_UPDATE = SIDE_NUMBER
THEN
     UPDATE_COMPLETED = YES
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, SIDENUM
! display current side number
     DISPLAY "THE AIRCRAFT SIDE NUMBER IS CURRENTLY
{SIDENUM}.
 "
     CLOSE FLIGHT
     RESET SIDENUM_UPDATE
! ask user for new aircraft side number
     FIND SIDENUM_UPDATE
     RESET SIDENUM_UPDATE_NOT_MISSING
! cannot allow a null value for side number
     FIND SIDENUM_UPDATE_NOT_MISSING
     RESET SIDENUM_UPDATE_EXISTS
! side number must match an aircraft in organization
     FIND SIDENUM_UPDATE_EXISTS
     RESET SIDENUM_UPDATE_EXISTS
     RESET SIDENUM_UPDATE_NOT_MISSING
     CLOSE FLIGHT
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, SIDENUM
     SIDENUM = (SIDE)
     PUT FLIGHT
     CLOSE FLIGHT
     FIELD_TO_UPDATE = DONE;

! *************** UPDATE EXCEPTION CODE **************

RULE UPDATE_EXCEPTION_CODE
IF
     FIELD_TO_UPDATE = EXCEPTION_CODE
THEN
     UPDATE_COMPLETED = YES
! display current exception code
     DISPLAY "THE EXCEPTION CODE IS CURRENTLY {EXCD}
                                                    "
     RESET EXCD_NEW
     FIND EXCD_NEW
! find if new exception code meets constraints
     RESET UPDATE_EXCD_VALID
```

```
     FIND UPDATE_EXCD_VALID
! from new exception code determine logic to keep database
! in a valid state
     RESET EXCD_RULE
     FIND EXCD_RULE;

! ********** EXCEPTION CODE LOGIC **********
! ********** RULE 1 **********
!     change exception code to X (canceled flight)

RULE EXCEPTION_RULE_1
IF
     FIELD_TO_UPDATE <> MISSION_1_CODE AND
     FIELD_TO_UPDATE = EXCEPTION_CODE AND
     EXCD_NEW = X AND
     EXCD_RULE = UNKNOWN
THEN
     EXCD_RULE = TRUE
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, EXCD
     RESET EXCD
     EXCD = (EXCD_NEW)
! need to get valid mission code 1
     RESET MSN1_1
     FIND MSN1_1
     RESET MSN11_VALID
     FIND MSN11_VALID
     CLS
     RESET MSN1_2
     FIND MSN1_2
     RESET MSN12_VALID
     FIND  MSN12_VALID
     CLS
     RESET MSN1_3
     FIND MSN1_3
     RESET MSN13_VALID
     FIND MSN13_VALID
     CLS
     RESET MSN11_VALID
     RESET MSN12_VALID
     RESET MSN13_VALID
! set all other flight attributes are null
     HRS1 = (BLANK)
     MSN2_1 = (BLANK)
     MSN2_2 = (BLANK)
     MSN2_3 = (BLANK)
     HRS2 = (BLANK)
     MSN3_1 = (BLANK)
     MSN3_2 = (BLANK)
     MSN3_3 = (BLANK)
     HRS3 = (BLANK)
     TOTFLT = (BLANK)
```

```
     OPS = (BLANK)
     CJ = (BLANK)
     MISNUM = (BLANK)
     NUMHOISTS = (BLANK)
     REMARKS = (BLANK)
     PUT FLIGHT
     CLOSE FLIGHT
! loop to remove related ENGINE records
     GET DOCNUM_UPDATE = DOCNUM, FLTENG, ALL
     WHILETRUE ENGNUM <> UNKNOWN THEN
          DOCNUM = (BLANK)
          ENGNUM = (BLANK)
          ENGHRS = (BLANK)
          PUT FLTENG
          GET DOCNUM_UPDATE = DOCNUM, FLTENG, ALL
     END
     CLOSE FLTENG
     FIELD_TO_UPDATE = DONE;

! *************** EXCEPTION CODE LOGIC **************
! ************************ RULE 2 *********************
! change exception code from X (canceled flight)

RULE EXCEPTION_RULE_2
IF
     FIELD_TO_UPDATE = EXCEPTION_CODE AND
     EXCD_RULE = UNKNOWN AND
     EXCD_NEW <> X AND
     EXCD = X
THEN
     EXCD_RULE = TRUE
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, EXCD
     DOCNUM = (DOCNUM_UPDATE)
     RESET EXCD
     EXCD = (EXCD_NEW)
     RESET MSN1_1
     FIND MSN1_1
     RESET MSN11_VALID
     FIND MSN11_VALID
! find mission code 1 position 2
     CLS
     RESET MSN1_2
     FIND MSN1_2
     RESET MSN12_VALID
     FIND  MSN12_VALID
! find mission code 1 position 3
     CLS
     RESET MSN1_3
     FIND MSN1_3
     RESET MSN13_VALID
     FIND MSN13_VALID
```

```
    CLS
! find mission 1 hours
    RESET CHECK
    FIND CHECK
    TEMPHRS1 = 0
    TEMPHRS2 = 0
    TEMPHRS3 = 0
    TOTHRS = 72.0
    SUBTOTAL = 0
    RESET HRS1_VALID
    FIND HRS1_VALID
! find mission code 2 position 1
    CLS
    RESET MSN21_VALID
    FIND MSN21_VALID
! find mission code 2 position 2
    CLS
    RESET MSN22_VALID
    FIND  MSN22_VALID
! find mission code 2 position 3
    CLS
    RESET MSN23_VALID
    FIND MSN23_VALID
    CLS
! find mission 2 hours
    RESET HRS2_VALID
    FIND HRS2_VALID
! find mission code 3 position 1
    CLS
    RESET MSN31_VALID
    FIND MSN31_VALID
! find mission code 3 position 2
    CLS
    RESET MSN32_VALID
    FIND  MSN32_VALID
! find mission code 3 position 3
    CLS
    RESET MSN33_VALID
    FIND MSN33_VALID
    CLS
! find mission 3 hours
    RESET HRS3_VALID
    FIND HRS3_VALID
    CLS
! find total flights
    RESET TOTFLT_VALID
    FIND TOTFLT_VALID
    CLS
! find ship/field operations code
    RESET OPS_VALID
    FIND OPS_VALID
```

```
     CLS
! find catapult/jato launches as necessary
     GET ALL, ORGAN, CATSJATO
     RESET CJ_VALID
     FIND CJ_VALID
     CLOSE ORGAN
     CLS
! find airlift mission number as necessary
     GET ALL, ORGAN, AIRLIFT
     RESET AIRLIFT_VALID
     FIND AIRLIFT_VALID
     CLOSE ORGAN
     CLS
! find number of hoists
     RESET NUMHOIST_VALID
     FIND NUMHOIST_VALID
! append new record to flight database
     PUT FLIGHT
     CLOSE FLIGHT
! find engine hours for aircraft on flight
     GET SIDE = (SIDENUM), AIRCRAFT, ENGINES
     CLOSE AIRCRAFT
     RESET UPDATE_ENGHRS_VALID
     FIND UPDATE_ENGHRS_VALID
     CLS
     FIELD_TO_UPDATE = DONE;

! ********** EXCEPTION CODE LOGIC **********
! ********** RULE 2 **********
! change exception code from a value not X to a value
! not X

RULE EXCEPTION_RULE_3
IF
     FIELD_TO_UPDATE = EXCEPTION_CODE AND
     EXCD_RULE = UNKNOWN AND
     EXCD_NEW <> X AND
     EXCD <> X
THEN
     EXCD_RULE = TRUE
     CLS
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     EXCD = (EXCD_NEW)
     PUT FLIGHT
     CLOSE FLIGHT
     CLS
     FIELD_TO_UPDATE = DONE;

! ********** UPDATE MISSION CODE 1 **********

RULE UPDATE_MISSION_CODE_1
```

```
IF
     FIELD_TO_UPDATE = MISSION_CODE_1
THEN
     UPDATE_COMPLETED = YES
! display current mission code 1
     DISPLAY "THE MISSION NUMBER 1 CODE IS CURRENTLY
{MSN1_1}{MSN1_2}{MSN1_3}




                          PRESS ANY KEY TO CONTINUE~"

! find mission code 1 position 1
     CLS
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, EXCD
     RESET EXCD_VALUE
     FIND EXCD_VALUE
! find mission code 1 position 1
     RESET MSN1_1
     FIND MSN1_1
     RESET MSN11_VALID
     FIND MSN11_VALID
! find mission code 1 position 2
     CLS
     RESET MSN1_2
     FIND MSN1_2
     RESET MSN12_VALID
     FIND  MSN12_VALID
! find mission code 1 position 3
     CLS
     RESET MSN1_3
     FIND MSN1_3
     RESET MSN13_VALID
     FIND MSN13_VALID
     CLS
! from new mission code 1 determine logic to keep database
! in a valid state
     RESET MISSION_CODE1_RULE
     FIND MISSION_CODE1_RULE;

! *************** MISSION CODE 1 LOGIC **************
! ************************ RULE 1 ********************
!                   exception code is X

RULE MISSION_CODE_1_RULE_1
IF
     FIELD_TO_UPDATE = MISSION_CODE_1 AND
     EXCD = X
```

```
THEN
     MISSION_CODE1_RULE = USED
     PUT FLIGHT
     CLOSE FLIGHT
     FIELD_TO_UPDATE = DONE;

! *************** MISSION CODE 1 LOGIC **************
! ************************ RULE 2 ********************
! mission code 1 position 1 is 6 and exception code
!                        is not X

RULE MISSION_CODE_1_RULE_2
IF
     FIELD_TO_UPDATE = MISSION_CODE_1 AND
     MSN1_1 = 6 AND
     EXCD <> X
THEN
     MISSION_CODE1_RULE = USED
     CLS
! find mission 1 hours
     RESET CHECK
     FIND CHECK
     TOTAL = 0
     RESET HRS1_VALID
     FIND HRS1_VALID
     RESET MSN2_1
! no other mission codes allowed
     MSN2_1 = (BLANK)
     RESET MSN2_2
     MSN2_2 = (BLANK)
     RESET MSN2_3
     MSN2_3 = (BLANK)
     RESET HRS2
     HRS2 = (BLANK)
     RESET MSN3_1
     MSN3_1 = (BLANK)
     RESET MSN3_2
     MSN3_2 = (BLANK)
     RESET MSN3_3
     MSN3_3 = (BLANK)
     RESET HRS3
     HRS3 = (BLANK)
     PUT FLIGHT
     CLOSE FLIGHT
! update engine hours for aircraft
     RESET MISSION1_ENGHRS_VALID
     FIND MISSION1_ENGHRS_VALID
     FIELD_TO_UPDATE = DONE;

! loop to update engine hours resulting from updating
! mission code 1 when mission code 1 position 1 is 6
```

```
! and exception code is not equal to X

RULE UPDATE_MISSION_1_ENGINE_HOURS
IF
     MSN1_1 = 6 AND
     EXCD <> X AND
     MISSION1_ENGHRS_VALID = UNKNOWN AND
     FIELD_TO_UPDATE = MISSION_CODE_1
THEN
     MISSION1_ENGHRS_VALID = TRUE
     Y = 1
     GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y), FLTENG,
ALL       WHILETRUE ENGNUM <> UNKNOWN THEN
           RESET ENGHRS
           FIND ENGHRS
           RESET ENGHRS_VALID
           RESET ENGHRS_LOOP
           FIND ENGHRS_LOOP
           Y = (Y +1)
           PUT FLTENG
           CLS
           GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
           FLTENG, ALL
          END
     CLOSE FLTENG;

! *************** MISSION CODE 1 LOGIC **************
! ************************ RULE 3 ********************
! mission code 1 position 1 is not 6 and exception code
!                        is not X

RULE MISSION_CODE_1_RULE_3
IF
     FIELD_TO_UPDATE = MISSION_CODE_1 AND
     MSN1_1 <> 6 AND
     EXCD <> X
THEN
     MISSION_CODE1_RULE = USED
     PUT FLIGHT
     CLOSE FLIGHT
     FIELD_TO_UPDATE = DONE;

! *************** UPDATE MISSION HOURS 1 ***************

RULE UPDATE_HRS1
IF
     FIELD_TO_UPDATE = MISSION_1_HOURS
THEN
     UPDATE_COMPLETED = YES
     TOTAL = 0
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
```

```
     RESET EXCD_VALUE
     FIND EXCD_VALUE
! find if mission hours 1 is valid
     RESET UPDATE_HRS1_VALID
     FIND UPDATE_HRS1_VALID
     PUT FLIGHT
     CLOSE FLIGHT
!  loop to update ENGINE records after change to mission
!  hours 1
     GET SIDENUM = SIDE, AIRCRAFT, ENGINES
     CLOSE AIRCRAFT
     Y = 1
     WHILETRUE UPDATE_HRS1_VALID <> FALSE AND Y <= (ENGINES)
     THEN
          GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
          FLTENG, ALL
          RESET ENGHRS
          FIND ENGHRS
! find if engine hours is valid
          RESET ENGHRS_VALID
          RESET ENGHRS_LOOP
          FIND ENGHRS_LOOP
          Y = (Y +1)
          PUT FLTENG
          CLS
     END
     CLOSE FLTENG
     FIELD_TO_UPDATE = DONE;

! *************** UPDATE MISSION CODE 2 ***************

RULE UPDATE_MISSION_CODE_2
IF
     FIELD_TO_UPDATE = MISSION_CODE_2
THEN
     UPDATE_COMPLETED = YES
! display current mission code 2
     DISPLAY "THE MISSION NUMBER 2 CODE IS CURRENTLY
{1MSN2_1}{1MSN2_2}{1MSN2_3}




                         PRESS ANY KEY TO CONTINUE~"

! find mission code 2 position 1
     CLS
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     CLOSE FLIGHT
```

```
     TOTAL = (HRS1)
! determine if you are allowed to update mission code 2
     RESET MISSION_CODE2_RULE
     FIND MISSION_CODE2_RULE
! from new mission code 2 determine logic to keep database
! in a valid state
     RESET MISSION_2_VALUE
     FIND MISSION_2_VALUE
     FIELD_TO_UPDATE = DONE;

! ********** MISSION CODE 2 ALLOWED ***********
! mission code 2 not allowed if exception code is X
!    or mission code 1 position 1 is equal to 6

RULE MISSION_CODE_2_RULE
IF
     FIELD_TO_UPDATE = MISSION_CODE_2 AND
     EXCD = X OR
     MSN1_1 = 6
THEN
! mission code 2 not allowed
! display message
     MISSION_CODE2_RULE = NOT_USED
     DISPLAY " YOU ARE NOT ALLOWED TO ENTER A MISSION CODE
                    FOR ONE OF THE FOLLOWING REASONS:
                      1. EXCEPTION CODE = X
                      2. MISSION CODE 1 BEGINS WITH A 6


                      PRESS ANY KEY TO CONTINUE

                              ~"
     CLS
     CLOSE FLIGHT
ELSE
! mission code 2 allowed
     MISSION_CODE2_RULE = USED
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
! find mission code 2 position 1
     RESET MSN2_1
     RESET MSN2_2
     RESET MSN2_3
     RESET MSN21_VALID
     FIND MSN21_VALID
! find mission code 2 position 2
     CLS
     RESET MSN22_VALID
     FIND  MSN22_VALID
! find mission code 2 position 3
     CLS
     RESET MSN23_VALID
```

```
     FIND MSN23_VALID
     PUT FLIGHT
     CLOSE FLIGHT;

! *************** MISSION CODE 2 LOGIC **************
! ************************ RULE 1 ********************
!                  new mission code 2 is null

RULE MISSION_CODE2_VALUE_RULE1
IF
     FIELD_TO_UPDATE = MISSION_CODE_2 AND
     MISSION_CODE2_RULE = USED AND
     SKIP = YES
THEN
     MISSION_2_VALUE = MISSING
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
! remove mission code 2, 3 along with mission hours 2, 3
     RESET MSN2_2
     MSN2_2 = (BLANK)
     RESET MSN2_3
     MSN2_3 = (BLANK)
     RESET HRS2
     HRS2 = (BLANK)
     RESET MSN3_1
     MSN3_1 = (BLANK)
     RESET MSN3_2
     MSN3_2 = (BLANK)
     RESET MSN3_3
     MSN3_3 = (BLANK)
     RESET HRS3
     HRS3 = (BLANK)
     PUT FLIGHT
     CLOSE FLIGHT
     Y = 1
! loop to update ENGINE records
     GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y), FLTENG,
     ALL
          WHILETRUE ENGNUM <> UNKNOWN THEN
           RESET ENGHRS
           FIND ENGHRS
           RESET ENGHRS_VALID
           RESET ENGHRS_LOOP
           FIND ENGHRS_LOOP
           Y = (Y +1)
           PUT FLTENG
           CLS
           GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
           FLTENG, ALL
          END
     CLOSE FLTENG;
```

```
! *************** MISSION CODE 2 LOGIC **************
! ************************ RULE 2 ********************
!            replace current mission code 2

RULE MISSION_CODE2_VALUE_RULE2
IF
     FIELD_TO_UPDATE = MISSION_CODE_2 AND
     MISSION_CODE2_RULE = USED AND
     TOTAL <> (HRS1 + HRS2 + HRS3) and
     SKIP = NO
THEN
     MISSION_2_VALUE = NOT_MISSING
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     PUT FLIGHT
     CLOSE FLIGHT;

! *************** MISSION CODE 2 LOGIC **************
! ************************ RULE 3 ********************
!          mission code 2 was previously null

RULE MISSION_CODE2_VALUE_RULE3
IF
     FIELD_TO_UPDATE = MISSION_CODE_2 AND
     MISSION_CODE2_RULE = USED AND
     TOTAL = (HRS1) AND
     SKIP = NO
THEN
     MISSION_2_VALUE = NOT_MISSING
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     CLS
     RESET CHECK
     FIND CHECK
     TEMPHRS1 = 0
     TEMPHRS2 = 0
     TEMPHRS3 = 0
     TOTHRS = 72.0
     SUBTOTAL = 0
! find mission 2 hours
     RESET HRS2
     FIND HRS2
     RESET TEST_HRS2
     WHILETRUE TEST_HRS2 = UNKNOWN  OR TEST_HRS2 = NOT_TRUE
     THEN
! find if mission hours 2 is valid
          RESET TEST_HRS2
          FIND TEST_HRS2
     END
     CLS
     PUT FLIGHT
     CLOSE FLIGHT
! loop to update ENGINE records
```

```
     Y = 1
     GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y), FLTENG,
     ALL
      WHILETRUE ENGNUM <> UNKNOWN THEN
          RESET ENGHRS
          FIND ENGHRS
! find if engine hours is valid
          RESET ENGHRS_VALID
          RESET ENGHRS_LOOP
          FIND ENGHRS_LOOP
          Y = (Y +1)
          PUT FLTENG
          CLS
          GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
          FLTENG, ALL
      END
     CLOSE FLTENG;

! *************** MISSION CODE 2 LOGIC **************
! ************************ RULE 4 ********************
!               mission code 2 is not allowed

RULE MISSION_CODE2_VALUE_RULE4
IF
     FIELD_TO_UPDATE = MISSION_CODE_2 AND
     MISSION_CODE2_RULE = NOT_USED
THEN
     MISSION_2_VALUE = NOT_REQUIRED
     CLS;

! *************** UPDATE MISSION HOURS 2 ***************

RULE UPDATE_HRS2
IF
     FIELD_TO_UPDATE = MISSION_2_HOURS
THEN
     UPDATE_COMPLETED = YES
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     TOTAL = (HRS1)
     CHECKSUM = (HRS1 +HRS2)
     RESET EXCD_VALUE
     FIND EXCD_VALUE
! find if mission hours 2 is valid
     RESET UPDATE_HRS2_VALID
     FIND UPDATE_HRS2_VALID
     PUT FLIGHT
     CLOSE FLIGHT
! loop to update ENGINE records
     Y = 1
     GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y), FLTENG,
     ALL
```

```
     WHILETRUE ENGNUM <> UNKNOWN AND UPDATE_HRS2_VALID <>
     FALSE
     THEN
          RESET ENGHRS
          FIND ENGHRS
          RESET ENGHRS_VALID
          RESET ENGHRS_LOOP
          FIND ENGHRS_LOOP
          Y = (Y +1)
          PUT FLTENG
          CLS
          GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
          FLTENG, ALL
     END
     CLOSE FLTENG
     FIELD_TO_UPDATE = DONE;

! ********** UPDATE MISSION CODE 2 **********

RULE UPDATE_MISSION_CODE_3
IF
     FIELD_TO_UPDATE = MISSION_CODE_3
THEN
     UPDATE_COMPLETED = YES
! display current mission code 3
     DISPLAY "THE MISSION NUMBER 3 CODE IS CURRENTLY
{1MSN3_1}{1MSN3_2}{1MSN3_3}




                         PRESS ANY KEY TO CONTINUE~"

! find mission code 3 position 1
     CLS
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     CLOSE FLIGHT
     TOTAL1 = (HRS1)
     TOTAL2 = (HRS1 + HRS2)
     TOTAL = (TOTAL2)
! determine if you are allowed to update mission code 3
     RESET MISSION_CODE3_RULE
     FIND MISSION_CODE3_RULE
! from new mission code 3 determine logic to keep database
! in a valid state
     RESET MISSION_3_VALUE
     FIND MISSION_3_VALUE
     FIELD_TO_UPDATE = DONE;
```

```
! *************** MISSION CODE 3 ALLOWED ************
! mission code 3 not allowed if exception code is X
!    or mission code 1 position 1 is equal to 6 or
!           mission code 2 is null

RULE MISSION_CODE_3_RULE
IF
     FIELD_TO_UPDATE = MISSION_CODE_3 AND
     EXCD = X OR
     MSN1_1 = 6 OR
     TOTAL2 = (TOTAL1)
THEN
! mission code 3 is not allowed
! display message
     MISSION_CODE3_RULE = NOT_USED
     DISPLAY " YOU ARE NOT ALLOWED TO ENTER A MISSION CODE
FOR ONE OF THE FOLLOWING REASONS:                          1.
EXCEPTION CODE = X
                    2. MISSION CODE 1 BEGINS WITH A 6
                    3. THERE IS NO MISSION 2 CODE

                    PRESS ANY KEY TO CONTINUE

                          ~"
     CLS
     CLOSE FLIGHT
ELSE
! mission code 3 is allowed
     MISSION_CODE3_RULE = USED
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
! find mission code 3 position 1
     RESET MSN3_1
     RESET MSN3_2
     RESET MSN3_3
     RESET MSN31_VALID
     FIND MSN31_VALID
! find mission code 3 position 2
     CLS
     RESET MSN32_VALID
     FIND  MSN32_VALID
! find mission code 3 position 3
     CLS
     RESET MSN33_VALID
     FIND MSN33_VALID
     PUT FLIGHT
     CLOSE FLIGHT;

! *************** MISSION CODE 3 LOGIC **************
! ************************ RULE 1 *********************
!               mission code 3 is null
```

```
RULE MISSION_CODE3_VALUE_RULE1
IF
     FIELD_TO_UPDATE = MISSION_CODE_3 AND
     MISSION_CODE3_RULE = USED AND
     SKIP_AGAIN = YES
THEN
     MISSION_3_VALUE = MISSING
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     RESET MSN3_2
! all related flight attributes are null
     MSN3_2 = (BLANK)
     RESET MSN3_3
     MSN3_3 = (BLANK)
     RESET HRS3
     HRS3 = (BLANK)
     RESET MSN3_1
     PUT FLIGHT
     CLOSE FLIGHT
! loop to update ENGINE records
     Y = 1
     GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y), FLTENG,
     ALL
     WHILETRUE ENGNUM <> UNKNOWN THEN
          RESET ENGHRS
          FIND ENGHRS
          RESET ENGHRS_VALID
          RESET ENGHRS_LOOP
          FIND ENGHRS_LOOP
          Y = (Y +1)
          PUT FLTENG
          CLS
          GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
          FLTENG, ALL
     END
     CLOSE FLTENG;

! ********** MISSION CODE 3 LOGIC **********
! ********** RULE 2 **********
!          replace current mission code 3

RULE MISSION_CODE3_VALUE_RULE2
IF
     FIELD_TO_UPDATE = MISSION_CODE_3 AND
     MISSION_CODE3_RULE = USED AND
     TOTAL2 <> (HRS1 + HRS2 + HRS3) and
     SKIP_AGAIN = NO
THEN
     MISSION_3_VALUE = NOT_MISSING
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     PUT FLIGHT
     CLOSE FLIGHT;
```

```
! *************** MISSION CODE 3 LOGIC **************
! ************************ RULE 2 ********************
!           mission code 3 was previously null

RULE MISSION_CODE3_VALUE_RULE3
IF
     FIELD_TO_UPDATE = MISSION_CODE_3 AND
     MISSION_CODE3_RULE = USED AND
     TOTAL2 = (HRS1 + HRS2 +HRS3) AND
     SKIP = NO
THEN
     MISSION_3_VALUE = NOT_MISSING
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     CLS
! find mission 3 hours
     RESET CHECK
     FIND CHECK
     TEMPHRS1 = 0
     TEMPHRS2 = 0
     TEMPHRS3 = 0
     TOTHRS = 72.0
     SUBTOTAL = 0
! find mission 3 hours
     RESET HRS3
     FIND HRS3
     RESET TEST_HRS3
     WHILETRUE TEST_HRS3 = UNKNOWN  OR TEST_HRS3 = NOT_TRUE
     THEN
         RESET TEST_HRS3
         FIND TEST_HRS3
     END
     CLS
     PUT FLIGHT
     CLOSE FLIGHT
! loop to update ENGINE records
     Y = 1
     GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y), FLTENG,
     ALL
     WHILETRUE ENGNUM <> UNKNOWN THEN
          RESET ENGHRS
          FIND ENGHRS
          RESET ENGHRS_VALID
          RESET ENGHRS_LOOP
          FIND ENGHRS_LOOP
          Y = (Y +1)
          PUT FLTENG
          CLS
          GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
          FLTENG, ALL
     END
     CLOSE FLTENG;
```

```
! *************** MISSION CODE 3 LOGIC **************
! ************************ RULE 2 ********************
!                mission code 3 not allowed

RULE MISSION_CODE3_VALUE_RULE4
IF
     FIELD_TO_UPDATE = MISSION_CODE_3 AND
     MISSION_CODE3_RULE = NOT_USED
THEN
     MISSION_3_VALUE = NOT_REQUIRED
     CLS;

! *************** UPDATE MISSION HOURS 3 ***************

RULE UPDATE_HRS3
IF
     FIELD_TO_UPDATE = MISSION_3_HOURS
THEN
     UPDATE_COMPLETED = YES
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
! display current mission hours 3
     DISPLAY "{HRS3} IS CURRENTLY THE MISSION 3 HOURS.
                                                    "
     TOTAL1 = (HRS1 + HRS2 + HRS3)
     TOTAL = (HRS1 + HRS2)
     CHECKSUM = (HRS1)
     RESET EXCD_VALUE
     FIND EXCD_VALUE
! find if mission hours 3 is valid
     RESET UPDATE_HRS3_VALID
     FIND UPDATE_HRS3_VALID
     PUT FLIGHT
     CLOSE FLIGHT
! loop to update ENGINE records
     Y = 1
     GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y), FLTENG,
     ALL
     WHILETRUE ENGNUM <> UNKNOWN AND UPDATE_HRS3_VALID <>
     FALSE THEN
          RESET ENGHRS
          FIND ENGHRS
          RESET ENGHRS_VALID
          RESET ENGHRS_LOOP
          FIND ENGHRS_LOOP
          Y = (Y +1)
          PUT FLTENG
          CLS
          GET DOCNUM_UPDATE = DOCNUM AND ENGNUM = (Y),
          FLTENG, ALL
     END
     CLOSE FLTENG
```

```
     FIELD_TO_UPDATE = DONE;

! ***************** UPDATE TOTAL FLIGHTS *****************

RULE UPDATE_TOTAL_FLIGHTS
IF
     FIELD_TO_UPDATE = TOTAL_FLIGHTS
THEN
     UPDATE_COMPLETED = YES
! display current total flight
     DISPLAY "{TOTFLT} IS CURRENTLY THE TOTAL FLIGHTS.
                                                      "
     CLOSE FLIGHT
! find if total flight is valid
     RESET UPDATE_TOTFLT_VALID
     FIND UPDATE_TOTFLT_VALID
     CLS
     FIELD_TO_UPDATE = DONE;

! ************ UPDATE SHIP/FIELD OPERATIONS **************

RULE UPDATE_SHIP_FIELD_OPERATIONS_CODE
IF
     FIELD_TO_UPDATE = SHIP_FIELD_OPERATIONS_CODE
THEN
     UPDATE_COMPLETED = YES
! display current ship/field operations code
     DISPLAY "{OPS} IS CURRENTLY THE SHIP/FIELD OPERATIONS
CODE.                                                 "
     CLOSE FLIGHT
! find if ship/field operations code is valid
     RESET UPDATE_OPS_VALID
     FIND UPDATE_OPS_VALID
     CLS
     FIELD_TO_UPDATE = DONE;

! ************ UPDATE CATAPULT/JATO LAUNCHES **************

RULE UPDATE_CATAPULT_JATO_LAUNCHES
IF
     FIELD_TO_UPDATE = CATAPULT_JATO_LAUNCHES
THEN
     UPDATE_COMPLETED = YES
! display current catapult/jato launches
     DISPLAY "{CJ} IS CURRENTLY THE NUMBER OF CATAPULT/JATO
LAUNCHES.
"
     CLOSE FLIGHT
! find catapult/jato launches as necessary
     GET ALL, ORGAN, CATSJATO
! find if cj is valid
```

```
     RESET UPDATE_CJ_VALID
     FIND UPDATE_CJ_VALID
     CLOSE ORGAN
     CLS
     FIELD_TO_UPDATE = DONE;

! ************ UPDATE AIRLIFT MISSION NUMBER **************

RULE UPDATE_AIRLIFT_MISSION_NUMBER
IF
     FIELD_TO_UPDATE = AIRLIFT_MISSION_NUMBER
THEN
     UPDATE_COMPLETED = YES
! display current airlift mission number
     DISPLAY "{MISNUM} IS CURRENTLY THE AIRLIFT MISSION
NUMBER.
"
     CLOSE FLIGHT
! find airlift mission number as necessary
     GET ALL, ORGAN, AIRLIFT
! find if airlift mission number is valid
     RESET UPDATE_AIRLIFT_VALID
     FIND UPDATE_AIRLIFT_VALID
     CLOSE ORGAN
     CLS
     FIELD_TO_UPDATE = DONE;

! ************ UPDATE NUMBER OF HOISTS **************

RULE UPDATE_NUMBER_OF_HOISTS
IF
     FIELD_TO_UPDATE = NUMBER_OF_HOISTS
THEN
     UPDATE_COMPLETED = YES
! display current number of hoists
     DISPLAY "{NUMHOISTS} IS CURRENTLY THE NUMBER OF HOISTS.
                                                          "
     CLOSE FLIGHT
! find if number of hoists is valid
     RESET UPDATE_NUMHOISTS_VALID
     FIND UPDATE_NUMHOISTS_VALID
     CLS
     FIELD_TO_UPDATE = DONE;

! ************ UPDATE ENGINE HOURS **************

RULE UPDATE_ENGINE_HOURS
IF
     FIELD_TO_UPDATE = ENGINE_HOURS
THEN
     UPDATE_COMPLETED = YES
```

```
! find if engine hours valid
     RESET UPDATE_ENGINE_HOURS_VALID
     FIND UPDATE_ENGINE_HOURS_VALID
     CLS
     FIELD_TO_UPDATE = DONE;

! ********************** UPDATE DONE ************************

RULE DONE
IF
     FIELD_TO_UPDATE = DONE
THEN
     UPDATE_COMPLETED = YES;

! **********************DELETE OPERATION**********************

RULE DELETE_DOCUMENT
IF
     WHICHTASK = DELETE_RECORD
THEN
     TASKCOMPLETED = YES
     MENU DOCNUM_DELETE, ALL, FLIGHT, DOCNUM
! ask user for document number from menu of document numbers
     FIND DOCNUM_DELETE
     MRESET DOCNUM_DELETE
! ask user to confirm delete operation
     RESET CONTINUE
     FIND CONTINUE
! find if any documents to delete
     RESET DELETE
     FIND DELETE;

! determine if there are any flight records to delete

RULE DELETE
IF
     DOCNUM_DELETE = NONE OR
     CONTINUE = NO AND
     DELETE = UNKNOWN
THEN
! no records to delete or user has changed mind
! display message
     DELETE = NO
     DISPLAY "      NO FLIGHT RECORD DELETED.

                    PRESS ANY KEY TO CONTINUE
                                ~"
     CLS
ELSE
! records available to delete
! and user has confirmed deletion
```

```
     DELETE = YES
     GET DOCNUM_DELETE = DOCNUM, FLIGHT, ALL
! all attributes are set to null
     DOCNUM = (BLANK)
     EXCD = (BLANK)
     MSN1_1 = (BLANK)
     MSN1_2 = (BLANK)
     MSN1_3 = (BLANK)
     HRS1 = (BLANK)
     MSN2_1 = (BLANK)
     MSN2_2 = (BLANK)
     MSN2_3 = (BLANK)
     HRS2 = (BLANK)
     MSN3_1 = (BLANK)
     MSN3_2 = (BLANK)
     MSN3_3 = (BLANK)
     HRS3 = (BLANK)
     TOTFLT = (BLANK)
     OPS = (BLANK)
     CJ = (BLANK)
     MISNUM = (BLANK)
     NUMHOISTS = (BLANK)
     REMARKS = (BLANK)
     SIDENUM = (BLANK)
     PUT FLIGHT
     CLOSE FLIGHT
! Cascade delete feature
! all associated records in with FLIGHT set to null
     GET DOCNUM_DELETE = DOCNUM, FLTENG, ALL
     WHILETRUE ENGNUM <> UNKNOWN THEN
          DOCNUM = (BLANK)
          ENGNUM = (BLANK)
          ENGHRS = (BLANK)
          PUT FLTENG
          GET DOCNUM_DELETE = DOCNUM, FLTENG, ALL
     END
     CLOSE FLTENG;

! ********************** VIEW OPERATION **********************

RULE VIEW_DOCUMENT
IF
     WHICHTASK = DISPLAY_RECORD
THEN
     RESET ALL
     WHICHTASK = DISPLAY_RECORD
     FORMAT HRS1, 4.1
     FORMAT HRS2, 4.1
     FORMAT HRS3, 4.1
     FORMAT ENGHRS, 4.1
     TASKCOMPLETED = YES
```

```
     RESET DOCNUM_VIEW
     MENU DOCNUM_VIEW, ALL, FLIGHT, DOCNUM
! ask user for document number
     FIND DOCNUM_VIEW
     MRESET DOCNUM_VIEW
! find if any flight records to view
     RESET VIEW
     FIND VIEW;

! determine if there are any documents to view

RULE VIEW
IF
     DOCNUM_VIEW = NONE AND
     VIEW = UNKNOWN
THEN
! no flight records to view
     VIEW = NO
     DISPLAY " THERE IS NO FLIGHT RECORD TO VIEW.

                    PRESS ANY KEY TO CONTINUE
                                    ~"
     CLS
ELSE
! flight record available to view
     VIEW = YES
     GET DOCNUM_VIEW = DOCNUM, FLIGHT, ALL
     SIDENO = (SIDENUM)
     CLOSE FLIGHT
     GET SIDENO = SIDE, AIRCRAFT, ALL
     CLOSE AIRCRAFT
     GET ALL, ORGAN, ALL
     CLOSE ORGAN
     CLS
! format for display
     DISPLAY "                         NAVAL AIRCRAFT FLIGHT RECORD

NO. {DOCNUM}
AIRCRAFT DATA
 {3SIDENUM} {1EXCD}  {6BUNO}    {4TEC} {3ORG}
{1MSN1_1}{1MSN1_2}{1MSN1_3}  {4HRS1}
{1MSN2_1}{1MSN2_2}{1MSN2_3}  {4HRS2}
{1MSN3_1}{1MSN3_2}{1MSN3_3}  {4HRS3}  {2SUPTCD}   {2TOTFLT}
{1OPS}   {2CJ} SIDE E BUNO/SER  TEC  ORG  MSN1 HRS1 MSN2
HRS2 MSN3 HRS3 SUPT  TOT  O  CAT/   NO. X
    TOTAL MISSION REQ DATA    CODE  FLT  P  JATO       C
                                                      S
D

    {9MISNUM}            {2NUMHOISTs}
AIRLIFT MISSION NO.       NO.
```

```
                        HOISTS
                                                                    "
    GET DOCNUM_VIEW = DOCNUM, FLTENG, ALL
    WHILEKNOWN ENGNUM
          DISPLAY " ENGINE NO{ENGNUM} ENGINE HOURS
          {4ENGHRS}"
          GET DOCNUM_VIEW = DOCNUM, FLTENG, ALL
    END
    DISPLAY "

                              PRESS ANY KEY TO CONTINUE....~"
    CLOSE FLTENG
    CLS;

! ********************** EXIT OPERATION ********************

RULE EXIT
IF
     WHICHTASK = EXIT
THEN
     TASKCOMPLETED = YES;

! rule to determine if exception code within record is
! null and assign value BLANK to it

RULE EXCEPTION_CODE_VALUE
IF
     WHICHTASK = UPDATE_RECORD AND
     EXCD = UNKNOWN
THEN
     EXCD_VALUE = NEEDED
     RESET EXCD
     EXCD = (BLANK)
ELSE
     EXCD_VALUE = NOT_NEEDED;

! rule to check value of exception code is equal to X
! and mission code 1 position 1 is equal to 6

RULE CHECK_VALUE
IF
     MSN1_1 = 6 OR
     EXCD = X
THEN
     CHECK = YES
ELSE
     CHECK = NO;

! rules to determine if repeating attributes
! (mission code n, n+1,...) should be skipped
```

```
RULE TEST_UNKNOWN1
IF
     MSN2_1 = UNKNOWN OR
     MSN2_1 = (BLANK)
THEN
     SKIP = YES
     SKIP_AGAIN = YES
ELSE
     SKIP = NO;

RULE TEST_UNKNOWN2
IF
     MSN3_1 = UNKNOWN OR
     MSN3_1 = (BLANK)
THEN
     SKIP_AGAIN = YES
ELSE
     SKIP_AGAIN = NO;

!************ KNOWLEDGE BASE LIBRARY *************

!********* DOMAIN INTEGRITY CONSTRAINTS *********

! THESE CONSTRAINTS ARE DEFINED IN DATA TYPES

!********* COLUMN INTEGRITY CONSTRAINTS *********

RULE EXCEPTION_CODE_VALID
IF
     EXCD_VALID = UNKNOWN
THEN
        WHILETRUE EXCD_VALID = UNKNOWN THEN
           RESET TEST_EXCD
           FIND TEST_EXCD
        END
        EXCD_VALID = TRUE;

RULE COLUMN_INTEGRITY_EXCEPTION_CODE
IF
     EXCD = C OR
     EXCD = D OR
     EXCD = X OR
     EXCD = UNKNOWN OR
     EXCD = (BLANK) AND
     EXCD_VALID = UNKNOWN
THEN
     TEST_EXCD = YES
     EXCD_VALID = TRUE
ELSE
     TEST_EXCD = YES
     DISPLAY " YOU NEED TO ENTER A VALID EXCEPTION CODE TO
```

```
     CONTINUE."
     RESET EXCD
     FIND EXCD;

RULE UPDATE_EXCEPTION_CODE_VALID
IF
     FIELD_TO_UPDATE = EXCEPTION_CODE AND
     UPDATE_EXCD_VALID = UNKNOWN
THEN
        WHILETRUE UPDATE_EXCD_VALID = UNKNOWN THEN
           RESET TEST_UPDATE_EXCD
           FIND TEST_UPDATE_EXCD
        END
        UPDATE_EXCD_VALID = TRUE;

RULE COLUMN_INTEGRITY_UPDATE_EXCEPTION_CODE
IF
     EXCD_NEW = C OR
     EXCD_NEW = D OR
     EXCD_NEW = X OR
     EXCD_NEW = (BLANK) AND
     FIELD_TO_UPDATE = EXCEPTION_CODE
THEN
     TEST_UPDATE_EXCD = YES
     UPDATE_EXCD_VALID = TRUE
ELSE
     TEST_UPDATE_EXCD = YES
     DISPLAY " YOU NEED TO ENTER A VALID EXCEPTION CODE TO
     CONTINUE."
     RESET EXCD_NEW
     FIND EXCD_NEW;

RULE MISSION_1_POSITION_1
IF
     MSN11_VALID = UNKNOWN
THEN
        WHILETRUE MSN11_VALID = UNKNOWN THEN
           RESET TEST_MSN11
           FIND TEST_MSN11
        END
        MSN11_VALID = TRUE;

RULE COLUMN_INTEGRITY_MISSION11_CODE
IF
     MSN1_1 >= 1 AND
     MSN1_1 <= 6
THEN
     TEST_MSN11 = YES
     MSN11_VALID = TRUE
ELSE
     TEST_MSN11 = YES
```

```
     DISPLAY " YOU MUST ENTER A NUMBER FROM 1 TO 6."
     RESET MSN1_1
     FIND MSN1_1;

RULE MISSION_1_POSITION_3
IF
     MSN13_VALID = UNKNOWN
THEN
        WHILETRUE MSN13_VALID = UNKNOWN THEN
           RESET TEST_MSN13
           FIND TEST_MSN13
        END
        MSN13_VALID = TRUE
        DISPLAY "{MSN1_1}{MSN1_2}{MSN1_3}
                 PRESS ANY KEY TO CONTINUE~";

RULE TEST_MISSION13_CODE_VALID
IF
     MSN1_3 >= 0 AND
     MSN1_3 <= 9
THEN
     TEST_MSN13 = YES
     MSN13_VALID = TRUE
ELSE
     TEST_MSN13 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 0 TO 9."
     RESET MSN1_3
     FIND MSN1_3;

RULE HRS_1_VALID
IF
     EXCD = X
THEN
     HRS1_VALID = NOT_NEEDED
ELSE
     RESET HRS1
     FIND HRS1
     WHILETRUE HRS1_VALID = UNKNOWN THEN
          RESET TEST_HRS1
          FIND TEST_HRS1
     END
     HRS1_VALID = TRUE;

RULE UPDATE_HRS_1_VALID
IF
     FIELD_TO_UPDATE = MISSION_1_HOURS AND
     EXCD = X
THEN
     DISPLAY " YOU ARE NOT ABLE TO ENTER HOURS FOR MISSION 1
               BECAUSE THE EXCEPTION CODE IS CURRENTLY {EXCD}
                                                        "
```

```
     UPDATE_HRS1_VALID = FALSE
ELSE
     SUBTOTAL_HOURS = ((HRS2) + (HRS3))
     ALLOWED_HOURS = (72 - (SUBTOTAL_HOURS))
     RESET HRS1
     FIND HRS1
     WHILETRUE UPDATE_HRS1_VALID = UNKNOWN THEN
          RESET UPDATE_TEST_HRS1
          FIND UPDATE_TEST_HRS1
     END
     UPDATE_HRS1_VALID = TRUE;

RULE UPDATE_HRS_2_VALID
IF
     FIELD_TO_UPDATE = MISSION_2_HOURS AND
     CHECK = YES OR
     TOTAL = (CHECKSUM)
THEN
     DISPLAY "YOU ARE NOT ABLE TO ENTER HOURS FOR MISSION 2
              BECAUSE:
                    1.  THE EXCEPTION CODE IS CURRENTLY X
                    2.  THE MISSION 1 CODE BEGINS WITH A 6
                    3.  THE MISSION 2 CODE IS MISSING
                                  "
     UPDATE_HRS2_VALID = FALSE
ELSE
     SUBTOTAL_HOURS = ((HRS1) + (HRS3))
     ALLOWED_HOURS = (72 - (SUBTOTAL_HOURS))
     RESET HRS2
     FIND HRS2
     WHILETRUE UPDATE_HRS2_VALID = UNKNOWN THEN
          RESET UPDATE_TEST_HRS2
          FIND UPDATE_TEST_HRS2
     END
     UPDATE_HRS2_VALID = TRUE;

RULE UPDATE_HRS_3_VALID
IF
     FIELD_TO_UPDATE = MISSION_3_HOURS AND
     CHECK = YES OR
     TOTAL = (CHECKSUM) OR
     TOTAL = (TOTAL1)
THEN
     DISPLAY "YOU ARE NOT ABLE TO ENTER HOURS FOR MISSION 3
              BECAUSE:
                    1.  THE EXCEPTION CODE IS CURRENTLY X
                    2.  THE MISSION 1 CODE BEGINS WITH A 6
                    3.  THE MISSION 2 CODE IS MISSING
                    4.  THE MISSION 3 CODE IS MISSING
                                                      "
      UPDATE_HRS3_VALID = FALSE
```

```
ELSE
     SUBTOTAL_HOURS = ((HRS1) + (HRS2))
     ALLOWED_HOURS = (72 - (SUBTOTAL_HOURS))
     RESET HRS3
     FIND HRS3
     WHILETRUE UPDATE_HRS3_VALID = UNKNOWN THEN
          RESET UPDATE_TEST_HRS3
          FIND UPDATE_TEST_HRS3
     END
     UPDATE_HRS3_VALID = TRUE;

RULE COLUMN_INTEGRITY_UPDATE_HRS1
IF
     FIELD_TO_UPDATE = MISSION_1_HOURS AND
     HRS1 > 0.0 AND
     HRS1 <= (ALLOWED_HOURS)
THEN
     UPDATE_TEST_HRS1 = YES
     UPDATE_HRS1_VALID = TRUE
     TOTAL = ((HRS1) + (SUBTOTAL_HOURS))
ELSE
     UPDATE_TEST_HRS1 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 00.1 TO
     {ALLOWED_HOURS}"
     RESET HRS1
     FIND HRS1;

RULE COLUMN_INTEGRITY_UPDATE_HRS2
IF
     FIELD_TO_UPDATE = MISSION_2_HOURS AND
     HRS2 > 0.0 AND
     HRS2 <= (ALLOWED_HOURS)
THEN
     UPDATE_TEST_HRS2 = YES
     UPDATE_HRS2_VALID = TRUE
     TOTAL = ((HRS2) + (SUBTOTAL_HOURS))
ELSE
     UPDATE_TEST_HRS2 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 00.1 TO
     {ALLOWED_HOURS}"
     RESET HRS2
     FIND HRS2;

RULE COLUMN_INTEGRITY_UPDATE_HRS3
IF
     FIELD_TO_UPDATE = MISSION_3_HOURS AND
     HRS3 > 0.0 AND
     HRS3 <= (ALLOWED_HOURS)
THEN
     UPDATE_TEST_HRS3 = YES
     UPDATE_HRS3_VALID = TRUE
```

```
     TOTAL = ((HRS3) + (SUBTOTAL_HOURS))
ELSE
     UPDATE_TEST_HRS3 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 00.1 TO
     {ALLOWED_HOURS}"
     RESET HRS3
     FIND HRS3;

RULE COLUMN_INTEGRITY_HRS1
IF
     HRS1 > 0.0 AND
     HRS1 <= 72.0
THEN
     TEST_HRS1 = YES
     HRS1_VALID = TRUE
     TOTAL = (HRS1)
ELSE
     TEST_HRS1 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 00.1 TO 72.0"
     RESET HRS1
     FIND HRS1;

RULE MISSION_2_POSITION_1
IF
     CHECK = YES
THEN
     MSN21_VALID = NOT_NEEDED
     SKIP = YES
     SKIP_AGAIN = YES
ELSE
        RESET MSN2_1
        FIND MSN2_1
        RESET SKIP
        FIND SKIP
        WHILETRUE MSN21_VALID = UNKNOWN THEN
           RESET TEST_MSN21
           FIND TEST_MSN21
        END
        MSN21_VALID = TRUE;

RULE COLUMN_INTEGRITY_MISSION21
IF
     MSN2_1 >= 1 OR
     MSN2_1 = UNKNOWN OR
     MSN2_1 = (BLANK) AND
     MSN2_1 <= 5 OR
     MSN2_1 = UNKNOWN OR
     MSN2_1 = (BLANK)
THEN
     TEST_MSN21 = YES
     MSN21_VALID = TRUE
```

```
ELSE
     TEST_MSN21 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 1 TO 5."
     RESET MSN2_1
     FIND MSN2_1;

RULE MISSION_2_POSITION_2
IF
     CHECK = YES OR
     SKIP = YES
THEN
     MSN22_VALID = NOT_NEEDED;

RULE TEST_MISSION23_CODE_VALID
IF
     CHECK = YES OR
     SKIP = YES
THEN
     MSN23_VALID = NOT_NEEDED
ELSE
     RESET MSN2_3
     FIND MSN2_3
     WHILETRUE MSN23_VALID = UNKNOWN THEN
           RESET TEST_MSN23
           FIND TEST_MSN23
     END
     MSN23_VALID = TRUE
     DISPLAY "{MSN2_1}{MSN2_2}{MSN2_3}
              PRESS ANY KEY TO CONTINUE~";

RULE TEST_MISSION23_CODE_VALID
IF
     MSN2_3 >= 0 AND
     MSN2_3 <= 9
THEN
     TEST_MSN23 = YES
     MSN23_VALID = TRUE
ELSE
     TEST_MSN23 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 0 TO 9."
     RESET MSN2_3
     FIND MSN2_3;

RULE HRS_2_VALID
IF
     CHECK = YES OR
     SKIP = YES
THEN
     HRS2_VALID = NOT_NEEDED
ELSE
     RESET HRS2
```

```
     FIND HRS2
     WHILETRUE HRS2_VALID = UNKNOWN THEN
         RESET TEST_HRS2
         FIND TEST_HRS2
     END
     HRS2_VALID = TRUE;

RULE COLUMN_INTEGRITY_HRS2
IF
     HRS2 > 0.0 AND
     HRS2 <= 72.0 AND
     TOTHRS >= (HRS1 + HRS2 + TEMPHRS3)
THEN
     TEST_HRS2 = YES
     HRS2_VALID = TRUE
     TOTAL = ((TOTAL) + (HRS2))
ELSE
     TEST_HRS2 = YES
     SUBTOTAL = (TOTHRS - HRS1)
     DISPLAY " YOU MUST ENTER A NUMBER FROM 00.1 TO
     {SUBTOTAL}"
     RESET HRS2
     FIND HRS2;

RULE MISSION_3_POSITION_1
IF
     CHECK = YES OR
     SKIP = YES
THEN
     MSN31_VALID = NOT_NEEDED
ELSE
        RESET MSN3_1
        FIND MSN3_1
        RESET SKIP_AGAIN
        FIND SKIP_AGAIN
        WHILETRUE MSN31_VALID = UNKNOWN THEN
           RESET TEST_MSN31
           FIND TEST_MSN31
        END
        MSN31_VALID = TRUE;

RULE COLUMN_INTEGRITY_MISSION31
IF
     MSN3_1 >= 1 OR
     MSN3_1 = UNKNOWN OR
     MSN3_1 = (BLANK) AND
     MSN3_1 <= 5 OR
     MSN3_1 = UNKNOWN OR
     MSN3_1 = (BLANK)
THEN
     TEST_MSN31 = YES
```

```
     MSN31_VALID = TRUE
ELSE
     TEST_MSN31 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 1 TO 5."
     RESET MSN3_1
     FIND MSN3_1;

RULE MISSION_3_POSITION_2
IF
     CHECK = YES OR
     SKIP = YES OR
     SKIP_AGAIN = YES
THEN
     MSN32_VALID = NOT_NEEDED;

RULE TEST_MISSION33_CODE_VALID
IF
     CHECK = YES OR
     SKIP = YES OR
     SKIP_AGAIN = YES
THEN
     MSN33_VALID = NOT_NEEDED
ELSE
     RESET MSN3_3
     FIND MSN3_3
     WHILETRUE MSN33_VALID = UNKNOWN THEN
          RESET TEST_MSN33
          FIND TEST_MSN33
     END
     MSN33_VALID = TRUE
     DISPLAY "{MSN3_1}{MSN3_2}{MSN3_3}
              PRESS ANY KEY TO CONTINUE~";

RULE TEST_MISSION33_CODE_VALID
IF
     MSN3_3 >= 0 AND
     MSN3_3 <= 9
THEN
     TEST_MSN33 = YES
     MSN33_VALID = TRUE
ELSE
     TEST_MSN33 = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 0 TO 9."
     RESET MSN3_3
     FIND MSN3_3;

RULE HRS_3_VALID
IF
     CHECK = YES OR
     SKIP = YES OR
     SKIP_AGAIN = YES
```

```
THEN
     HRS3_VALID = NOT_NEEDED
ELSE
     RESET HRS3
     FIND HRS3
     WHILETRUE HRS3_VALID = UNKNOWN THEN
         RESET TEST_HRS3
         FIND TEST_HRS3
     END
     HRS3_VALID = TRUE;

RULE COLUMN_INTEGRITY_HRS3
IF
     HRS3 > 0.0 AND
     HRS3 <= 72.0 AND
     TOTHRS >= (HRS1 + HRS2 + HRS3)
THEN
     TEST_HRS3 = YES
     HRS3_VALID = TRUE
     TOTAL = ((TOTAL) + (HRS3))
ELSE
     TEST_HRS3 = YES
     SUBTOTAL = (TOTHRS - (HRS1 + HRS2))
     DISPLAY " YOU MUST ENTER A NUMBER FROM 00.1 TO
     {SUBTOTAL}"
     RESET HRS3
     FIND HRS3;

RULE TOTAL_FLIGHTS_VALID
IF
     EXCD = X
THEN
     TOTFLT_VALID = NOT_NEEDED
ELSE
     RESET TOTFLT
     FIND TOTFLT
     WHILETRUE TOTFLT_VALID = UNKNOWN THEN
         RESET TEST_TOTFLT
         FIND TEST_TOTFLT
     END
     TOTFLT_VALID = TRUE;

RULE UPDATE_TOTAL_FLIGHTS_VALID
IF
     EXCD = X AND
     FIELD_TO_UPDATE = TOTAL_FLIGHTS
THEN
     UPDATE_TOTFLT_VALID = NOT_NEEDED
     DISPLAY " YOU ARE NOT ABLE TO ENTER TOTAL FLIGHTS FOR
     {DOCNUM_UPDATE} BECAUSE THE EXCEPTION CODE IS CURRENTLY
     {EXCD}
```

```
                    PRESS ANY KEY TO CONTINUE~"
     CLS
 ELSE
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, TOTFLT
     RESET TOTFLT
     FIND TOTFLT
     WHILETRUE UPDATE_TOTFLT_VALID = UNKNOWN THEN
         RESET TEST_TOTFLT
         FIND TEST_TOTFLT
     END
     PUT FLIGHT
     CLOSE FLIGHT
     UPDATE_TOTFLT_VALID = TRUE;


RULE COLUMN_INTEGRITY_TOTFLT
IF
     TOTFLT >= 1 AND
     TOTFLT <= 99
THEN
     TEST_TOTFLT = YES
     TOTFLT_VALID = TRUE
     UPDATE_TOTFLT_VALID = TRUE
ELSE
     TEST_TOTFLT = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 1 TO 99"
     RESET TOTFLT
     FIND TOTFLT;

RULE OPS_CODE_VALID
IF
     EXCD = X
THEN
     OPS_VALID = NOT_NEEDED
ELSE
        WHILETRUE OPS_VALID = UNKNOWN THEN
           RESET TEST_OPS
           FIND TEST_OPS
        END
        OPS_VALID = TRUE;

RULE UPDATE_SHIP_FIELD_OPS_CODE_VALID
IF
     EXCD = X AND
     FIELD_TO_UPDATE = SHIP_FIELD_OPERATIONS_CODE
THEN
     UPDATE_OPS_VALID = NOT_NEEDED
     DISPLAY "YOU ARE NOT ABLE TO ENTER SHIP/FIELD
              OPERATIONS CODE FOR {DOCNUM_UPDATE}
         BECAUSE THE EXCEPTION CODE IS CURRENTLY {EXCD}
```

```
                    PRESS ANY KEY TO CONTINUE~"
     CLS
 ELSE
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, OPS
     RESET OPS
     FIND OPS
     WHILETRUE UPDATE_OPS_VALID = UNKNOWN THEN
         RESET TEST_OPS
         FIND TEST_OPS
     END
     PUT FLIGHT
     CLOSE FLIGHT
     UPDATE_OPS_VALID = TRUE;

RULE COLUMN_INTEGRITY_OPS_CODE
IF
     OPS = A OR
     OPS = B OR
     OPS = 1 OR
     OPS = 2
THEN
     TEST_OPS = YES
     OPS_VALID = TRUE
     UPDATE_OPS_VALID = TRUE
ELSE
     TEST_OPS = YES
     DISPLAY " YOU NEED TO ENTER AN A, B, 1, OR 2."
     RESET OPS
     FIND OPS;

RULE CATS_JATO_VALID
IF
     CATSJATO = N OR
     EXCD = X
THEN
     CJ_VALID = NOT_NEEDED
ELSE
     RESET CJ
     FIND CJ
     WHILETRUE CJ_VALID = UNKNOWN THEN
         RESET TEST_CJ
         FIND TEST_CJ
     END
     CJ_VALID = TRUE;

RULE UPDATE_CATAPULT_JATO_LAUNCHES_VALID
IF
     CATSJATO = N OR
     EXCD = X AND
     FIELD_TO_UPDATE = CATAPULT_JATO_LAUNCHES
```

```
THEN
     UPDATE_CJ_VALID = NOT_NEEDED
     DISPLAY "YOU ARE NOT ABLE TO ENTER CATAPULT/JATO
             LAUNCHES FOR {DOCNUM_UPDATE} BECAUSE EITHER
     1. YOUR ORGANIZATION DOES NOT DOCUMENT CATAPULT/JATO
        LAUNCHES
     2. THE EXCEPTION CODE IS CURRENTLY X

                        PRESS ANY KEY TO CONTINUE~"
     CLS
 ELSE
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, CJ
     RESET CJ
     FIND CJ
     WHILETRUE UPDATE_CJ_VALID = UNKNOWN THEN
         RESET TEST_CJ
         FIND TEST_CJ
     END
     PUT FLIGHT
     CLOSE FLIGHT
     UPDATE_CJ_VALID = TRUE;

RULE COLUMN_INTEGRITY_CJ
IF
     CJ >= 1 OR
     CJ = UNKNOWN OR
     CJ = (BLANK) AND
     CJ <= 99 OR
     CJ = UNKNOWN OR
     CJ = (BLANK)
THEN
     TEST_CJ = YES
     CJ_VALID = TRUE
     UPDATE_CJ_VALID = TRUE
ELSE
     TEST_CJ = YES
     DISPLAY " YOU MUST ENTER A NUMBER FROM 1 TO 99 OR
                          <SPACE> FOR NONE"
     RESET CJ
     FIND CJ;

RULE AIRLIFT_MISSION_NUMBER_VALID
IF
     AIRLIFT = N OR
     EXCD = X
THEN
     AIRLIFT_VALID = NOT_NEEDED
ELSE
     RESET MISNUM
     FIND MISNUM
     AIRLIFT_VALID = TRUE;
```

```
RULE UPDATE_AIRLIFT_MISSION_NUMBER_VALID
IF
     AIRLIFT = N OR
     EXCD = X AND
     FIELD_TO_UPDATE = AIRLIFT_MISSION_NUMBER
THEN
     UPDATE_AIRLIFT_VALID = NOT_NEEDED
     DISPLAY "YOU ARE NOT ABLE TO ENTER AIRLIFT MISSION
              NUMBERS FOR {DOCNUM_UPDATE} BECAUSE EITHER
        1. YOUR ORGANIZATION DOES NOT DOCUMENT AIRLIFT
           MISSION NUMBERS
        2. THE EXCEPTION CODE IS CURRENTLY X

                    PRESS ANY KEY TO CONTINUE~"
     CLS
 ELSE
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, MISNUM
     RESET MISNUM
     FIND MISNUM
     PUT FLIGHT
     CLOSE FLIGHT
     UPDATE_AIRLIFT_VALID = TRUE;

RULE NUMHOIST_VALID
IF
     NUMHOIST_VALID = UNKNOWN AND
     EXCD = X
THEN
     NUMHOIST_VALID = NOT_NEEDED
ELSE
        WHILETRUE NUMHOIST_VALID = UNKNOWN THEN
           RESET TEST_NUMHOIST
           FIND TEST_NUMHOIST
        END
        NUMHOIST_VALID = TRUE;

RULE UPDATE_NUMHOISTS_VALID
IF
     EXCD = X AND
     FIELD_TO_UPDATE = NUMBER_OF_HOISTS
THEN
     UPDATE_NUMHOISTS_VALID = NOT_NEEDED
     DISPLAY "YOU ARE NOT ABLE TO ENTER NUMBER OF HOISTS FOR
              {DOCNUM_UPDATE} BECAUSE THE EXCEPTION CODE IS
              CURRENTLY {EXCD}

                         PRESS ANY KEY TO CONTINUE~"
     CLS
 ELSE
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, NUMHOISTS
     RESET NUMHOISTS
```

```
     FIND NUMHOISTS
     WHILETRUE UPDATE_NUMHOISTS_VALID = UNKNOWN THEN
         RESET TEST_NUMHOIST
         FIND TEST_NUMHOIST
     END
     PUT FLIGHT
     CLOSE FLIGHT
     UPDATE_NUMHOISTS_VALID = TRUE;


RULE COLUMN_INTEGRITY_NUMHOIST
IF
     NUMHOISTS >= 1 OR
     NUMHOISTS = UNKNOWN OR
     NUMHOISTS = (BLANK) AND
     NUMHOISTS <= 99 OR
     NUMHOISTS = UNKNOWN OR
     NUMHOISTS = (BLANK)
THEN
     TEST_NUMHOIST = YES
     NUMHOIST_VALID = TRUE
     UPDATE_NUMHOISTS_VALID = TRUE
ELSE
     TEST_NUMHOIST = YES
     DISPLAY " YOU NEED TO ENTER A NUMBER FROM 1 TO 99, OR ?
     FOR NONE."
     RESET NUMHOISTS
     FIND NUMHOISTS;

RULE UPDATE_ENGINE_HOURS_VALID
IF
     EXCD = X AND
     FIELD_TO_UPDATE = ENGINE_HOURS
THEN
     UPDATE_ENGINE_HOURS_VALID = NOT_NEEDED
     DISPLAY " YOU ARE NOT ABLE TO ENTER ENGINE HOURS FOR
               {DOCNUM_UPDATE} BECAUSE THE EXCEPTION CODE IS
               CURRENTLY {EXCD}

                         PRESS ANY KEY TO CONTINUE~"
     CLS
 ELSE
     UPDATE_ENGINE_HOURS_VALID = NEEDED
     GET DOCNUM_UPDATE = DOCNUM, FLIGHT, ALL
     TOTAL = (HRS1 + HRS2 + HRS3)
     CLOSE FLIGHT
     RESET ENGINE_NUMBER
     MENU ENGINE_NUMBER, DOCNUM_UPDATE = DOCNUM, FLTENG,
     ENGNUM
     FIND ENGINE_NUMBER
     MRESET ENGINE_NUMBER
```

```
      CLOSE FLTENG
      GET DOCNUM_UPDATE = DOCNUM AND ENGINE_NUMBER = ENGNUM,
      FLTENG, ENGHRS
      RESET UPDATE_ENGHRS
      FIND UPDATE_ENGHRS
      ENGHRS = (UPDATE_ENGHRS)
      RESET ENGHRS_VALID
      RESET ENGHRS_LOOP
      FIND ENGHRS_LOOP
      PUT FLTENG
      CLS
      CLOSE FLTENG
      FIELD_TO_UPDATE = DONE;

RULE COLUMN_INTEGRITY_ENGINE_HOURS
IF
      ENGHRS > 0 AND
      ENGHRS <= (TOTAL)
THEN
      TEST_ENGHRS = YES
      ENGHRS_VALID = TRUE
ELSE
      TEST_ENGHRS = YES
      DISPLAY " YOU MUST ENTER ENGINE HOURS BETWEEN 00.1 AND
      {4TOTAL}."
      RESET ENGHRS
      FIND ENGHRS;

!*********** ENTITY CONSTRAINT RULES ***********

RULE ENTITY_INTEGRITY_DOCNUM_MISSING
IF
      DOCNUM_NEW = UNKNOWN OR
      DOCNUM_NEW = (BLANK)
THEN
! loop to get user to enter a document number
      WHILETRUE DOCNUM_NEW = UNKNOWN OR DOCNUM_NEW = (BLANK)
      THEN
            DISPLAY " YOU WILL NOT BE ABLE TO PROCEED UNLESS
            YOU ENTER A DOCUMENT NUMBER."
            RESET DOCNUM_NEW
            FIND DOCNUM_NEW
      END
      DOCNUM_NOT_MISSING = TRUE
ELSE
      DOCNUM_NOT_MISSING = TRUE;

RULE ENTITY_INTEGRITY_DOCNUM_DUPLICATE
IF
      DOCNUM_NEW <> UNKNOWN OR
      DOCNUM_NEW <> (BLANK)
```

```
THEN
      GET DOCNUM_NEW = DOCNUM, FLIGHT, DOCNUM
      WHILETRUE DOCNUM = (DOCNUM_NEW) THEN
            CLOSE FLIGHT
            DISPLAY " THERE IS ALREADY A DOCUMENT NUMBER
            {DOCNUM_NEW} THAT EXISTS WITHIN THE DATABASE."
            RESET DOCNUM_NEW
! get another document number
            FIND DOCNUM_NEW
            RESET DOCNUM_NOT_MISSING
! once again must verify that document number is not a null
! value
             FIND DOCNUM_NOT_MISSING
             GET DOCNUM_NEW = DOCNUM, FLIGHT, DOCNUM
      END
      DOCNUM_DUPLICATE = FALSE;

RULE ENTITY_INTEGRITY_UPDATE_SIDENUM_MISSING
IF
      FIELD_TO_UPDATE = SIDE_NUMBER AND
      SIDENUM_UPDATE = UNKNOWN OR
      SIDENUM_UPDATE = (BLANK)
THEN
! loop to get user to enter a aircraft side number
      WHILETRUE SIDENUM_UPDATE = UNKNOWN OR SIDENUM_UPDATE =
      (BLANK) THEN
            DISPLAY " YOU WILL NOT BE ABLE TO PROCEED UNLESS
            YOU ENTER A AIRCRAFT SIDE NUMBER."
            RESET SIDENUM_UPDATE
            FIND SIDENUM_UPDATE
      END
      SIDENUM_UPDATE_NOT_MISSING = TRUE
ELSE
      SIDENUM_UPDATE_NOT_MISSING = TRUE;

RULE ENTITY_CONSTRAINT_SIDENUM_MISSING
IF    WHICHTASK = APPEND_RECORD AND
      SIDENUM_NEW = UNKNOWN OR
      SIDENUM_NEW = (BLANK)
THEN
! loop to get user to enter a aircraft side number
      WHILETRUE SIDENUM_NEW = UNKNOWN OR SIDENUM_NEW =
      (BLANK) THEN
            DISPLAY " YOU WILL NOT BE ABLE TO PROCEED UNLESS
            YOU ENTER A AIRCRAFT SIDE NUMBER."
            RESET SIDENUM_NEW
            FIND SIDENUM_NEW
      END
      SIDENUM_NOT_MISSING = TRUE
ELSE
      SIDENUM_NOT_MISSING = TRUE;
```

```
!*********** REFERENTIAL CONSTRAINT RULES ***********

RULE REFERENTIAL_INTEGRITY_SIDENUM_EXISTS
IF
     WHICHTASK = APPEND_RECORD AND
     SIDENUM_NEW <> UNKNOWN OR
     SIDENUM_NEW <> (BLANK)
THEN
     GET SIDENUM_NEW = SIDE, AIRCRAFT, SIDE
! loop till side number matches an aircraft in organization
   WHILETRUE SIDE = UNKNOWN THEN
          CLOSE AIRCRAFT
          DISPLAY " NO AIRCRAFT EXISTS IN THE ORGANIZATION
          WITH THE SIDE NUMBER {SIDENUM_NEW}."
          RESET SIDENUM_NEW
! get another side number
          FIND SIDENUM_NEW
          RESET SIDENUM_NOT_MISSING
! once again must verify that side number is not a null
! value
          FIND SIDENUM_NOT_MISSING
          GET SIDENUM_NEW = SIDE, AIRCRAFT, SIDE
    END
    SIDENUM_EXISTS = TRUE
    SIDENUM = (SIDE);


RULE REFERENTIAL_INTEGRITY_UPDATE_SIDENUM_EXISTS
IF
     FIELD_TO_UPDATE = SIDE_NUMBER AND
     SIDENUM_UPDATE <> UNKNOWN OR
     SIDENUM_UPDATE <> (BLANK)
THEN
     GET SIDENUM_UPDATE = SIDE, AIRCRAFT, SIDE
! loop till side number matches an aircraft in organization
   WHILETRUE SIDE = UNKNOWN THEN
          CLOSE AIRCRAFT
          DISPLAY " NO AIRCRAFT EXISTS IN THE ORGANIZATION
          WITH THE SIDE NUMBER {SIDENUM_UPDATE}."
          RESET SIDENUM_UPDATE
! get another side number
          FIND SIDENUM_UPDATE
          RESET SIDENUM_UPDATE_NOT_MISSING
! once again must verify that side number is not a null
! value
          FIND SIDENUM_UPDATE_NOT_MISSING
          GET SIDENUM_UPDATE = SIDE, AIRCRAFT, SIDE
    END
    SIDENUM_UPDATE_EXISTS = TRUE
    SIDENUM = (SIDE)
    CLOSE AIRCRAFT;
```

```
RULE ENGINE_HOURS_VALID
IF
     EXCD = X AND
     ENGHRS_VALID = UNKNOWN AND
     FIELD_TO_UPDATE <> ENGINE_HOURS
THEN
     ENGHRS_VALID = NOT_NEEDED
ELSE
     Y= 0
     ENGINE = (ENGINES - 1)
     WHILETRUE Y <= (ENGINE) THEN
          RESET ENGHRS_VALID
          ENGNUM = (Y +1)
          RESET ENGHRS
          FIND ENGHRS
          RESET ENGHRS_LOOP
          FIND ENGHRS_LOOP
          Y = (Y +1)
          APPEND FLTENG
          CLS
     END
     ENGHRS_VALID = TRUE;

RULE ENGINE_HOURS_LOOP
IF
     ENGHRS_LOOP = UNKNOWN
THEN
     WHILETRUE ENGHRS_VALID = UNKNOWN THEN
         RESET TEST_ENGHRS
         FIND TEST_ENGHRS
     END
     ENGHRS_LOOP = TRUE;

RULE UPDATE_ENGINE_HOURS_VALID_1
IF
     EXCD = X AND
     UPDATE_ENGHRS_VALID = UNKNOWN AND
     WHICHTASK = UPDATE_RECORD AND
     FIELD_TO_UPDATE <> ENGINE_HOURS
THEN
     UPDATE_ENGHRS_VALID = NOT_NEEDED;

RULE UPDATE_ENGINE_HOURS_VALID_2
IF
     EXCD <> X AND
     UPDATE_ENGHRS_VALID = UNKNOWN AND
     WHICHTASK = UPDATE_RECORD AND
     FIELD_TO_UPDATE <> ENGINE_HOURS
THEN
     Y = 0
     ENGINE = (ENGINES - 1)
```

```
      WHILETRUE Y <= (ENGINE) THEN
           RESET ENGHRS_VALID
           ENGNUM = (Y + 1)
           RESET ENGHRS
           FIND ENGHRS
           RESET ENGHRS_LOOP
           FIND ENGHRS_LOOP
           Y = (Y +1)
           APPEND FLTENG
           CLS
      END
      UPDATE_ENGHRS_VALID = TRUE;

RULE UPDATE_ENGINE_HOURS_LOOP
IF
      UPDATE_ENGHRS_LOOP = UNKNOWN
THEN
      WHILETRUE UPDATE_ENGHRS_VALID = UNKNOWN THEN
          RESET TEST_ENGHRS
          FIND TEST_ENGHRS
      END
      UPDATE_ENGHRS_LOOP = TRUE;

!************ USER DEFINED CONSTRAINT RULES ************

RULE MISSION_POSITION_12A
IF
       MSN1_1 = 1 AND
       EXCD <> X AND
       MSN12_VALID = UNKNOWN
THEN
          WHILETRUE MSN12_VALID = UNKNOWN THEN
             RESET CK_MSN12A
             FIND CK_MSN12A
             RESET REPEAT_REQUEST2A
             FIND REPEAT_REQUEST2A
          END
          MSN12_VALID = TRUE;

RULE USER_DEFINED_MISSION12A_CODE
IF
      MSN1_2 = A OR
      MSN1_2 = B OR
      MSN1_2 = C OR
      MSN1_2 = D OR
      MSN1_2 = E OR
      MSN1_2 = F OR
      MSN1_2 = G OR
      MSN1_2 = H OR
      MSN1_2 = I OR
      MSN1_2 = N OR
```

```
      MSN1_2 = O OR
      MSN1_2 = P OR
      MSN1_2 = R AND
      MSN1_1 = 1 AND
      EXCD <> X
THEN
      CK_MSN12A = YES
      MSN12_VALID = TRUE;

RULE USER_DEFINED_MISSION12AA_CODE
IF
      MSN1_2 <> A OR
      MSN1_2 <> B OR
      MSN1_2 <> C OR
      MSN1_2 <> D OR
      MSN1_2 <> E OR
      MSN1_2 <> F OR
      MSN1_2 <> G OR
      MSN1_2 <> H OR
      MSN1_2 <> I OR
      MSN1_2 <> N OR
      MSN1_2 <> O OR
      MSN1_2 <> P OR
      MSN1_2 <> R AND
      MSN1_1 = 1 AND
      EXCD <> X
THEN
      CK_MSN12A = YES;

RULE REPEAT_REQUEST_12A
IF
      MSN12_VALID <> UNKNOWN
THEN
      REPEAT_REQUEST2A = NO
ELSE
      CK_MSN12A = YES
      REPEAT_REQUEST2A = YES
      CLS
      DISPLAY " POSITION 2 MUST BE R, A-I, OR N-P.
                      PRESS ENTER TO CONTINUE. ~"
      CLS
      RESET MSN1_2
      FIND MSN1_2;

RULE MISSION_POSITION_12B
IF
       MSN1_1 = 2 AND
       EXCD <> X AND
       MSN12_VALID = UNKNOWN
THEN
         WHILETRUE MSN12_VALID = UNKNOWN THEN
```

```
            RESET CK_MSN12B
            FIND CK_MSN12B
            RESET REPEAT_REQUEST2B
            FIND REPEAT_REQUEST2B
         END
         MSN12_VALID = TRUE;

RULE USER_DEFINED_MISSION12B_CODE
IF
      MSN1_2 = J OR
      MSN1_2 = K OR
      MSN1_2 = L OR
      MSN1_2 = M OR
      MSN1_2 = N OR
      MSN1_2 = O OR
      MSN1_2 = P OR
      MSN1_2 = Q OR
      MSN1_2 = R AND
      MSN1_1 = 2 AND
      EXCD <> X
THEN
      CK_MSN12B = YES
      MSN12_VALID = TRUE;

RULE USER_DEFINED_MISSION12BB_CODE
IF
      MSN1_2 <> J OR
      MSN1_2 <> K OR
      MSN1_2 <> L OR
      MSN1_2 <> M OR
      MSN1_2 <> N OR
      MSN1_2 <> O OR
      MSN1_2 <> P OR
      MSN1_2 <> Q OR
      MSN1_2 <> R AND
      MSN1_1 = 2 AND
      EXCD <> X
THEN
      CK_MSN12B = YES;

RULE REPEAT_REQUEST_12B
IF
      MSN12_VALID <> UNKNOWN
THEN
      REPEAT_REQUEST2B = NO
ELSE
      REPEAT_REQUEST2B = YES
      CLS
      DISPLAY " POSITION 2 MUST BE IN THE RANGE OF J-R.
                        PRESS ENTER TO CONTINUE. ~"
      CLS
```

```
      RESET MSN1_2
      FIND MSN1_2;

RULE MISSION_POSITION_12C
IF
       MSN1_1 >= 3 AND
       EXCD <> X AND
       MSN12_VALID = UNKNOWN
THEN
         WHILETRUE MSN12_VALID = UNKNOWN THEN
            RESET CK_MSN12C
            FIND CK_MSN12C
            RESET REPEAT_REQUEST2C
            FIND REPEAT_REQUEST2C
         END
         MSN12_VALID = TRUE;

RULE USER_DEFINED_MISSION12C_CODE
IF
      MSN1_2 = N OR
      MSN1_2 = O OR
      MSN1_2 = S OR
      MSN1_2 = T OR
      MSN1_2 = U OR
      MSN1_2 = V OR
      MSN1_2 = W OR
      MSN1_2 = X OR
      MSN1_2 = Y OR
      MSN1_2 = Z AND
      MSN1_1 >= 3 AND
      EXCD <> X
THEN
      CK_MSN12C = YES
      MSN12_VALID = TRUE;

RULE USER_DEFINED_MISSION12CC_CODE
IF
      MSN1_2 <> N OR
      MSN1_2 <> O OR
      MSN1_2 <> S OR
      MSN1_2 <> T OR
      MSN1_2 <> U OR
      MSN1_2 <> V OR
      MSN1_2 <> W OR
      MSN1_2 <> X OR
      MSN1_2 <> Y OR
      MSN1_2 <> Z AND
      MSN1_1 >= 3 AND
      EXCD <> X
THEN
      CK_MSN12C = YES;
```

```
 RULE REPEAT_REQUEST_12C
IF
     MSN12_VALID <> UNKNOWN
THEN
     REPEAT_REQUEST2C = NO
ELSE
     REPEAT_REQUEST2C = YES
     CLS
     DISPLAY " POSITION 2 MUST BE N, O, OR S-Z.
                  PRESS ENTER TO CONTINUE. ~"
     CLS
     RESET MSN1_2
     FIND MSN1_2;

RULE MISSION_POSITION_12D
IF
      EXCD = X AND
      MSN12_VALID = UNKNOWN
THEN
        WHILETRUE MSN12_VALID = UNKNOWN THEN
           RESET CK_MSN12D
           FIND CK_MSN12D
           RESET REPEAT_REQUEST2D
           FIND REPEAT_REQUEST2D
        END
        MSN12_VALID = TRUE;

RULE USER_DEFINED_MISSION12D_CODE
IF
     MSN1_2 = N OR
     MSN1_2 = O AND
     EXCD = X
THEN
     CK_MSN12D = YES
     MSN12_VALID = TRUE;

RULE TEST_MISSION12DD_CODE_VALID
IF
     MSN1_2 <> N AND
     EXCD = X
THEN
     CK_MSN12D = YES;

RULE TEST_MISSION12DDD_CODE_VALID
IF
     MSN1_2 <> O AND
     EXCD = X
THEN
     CK_MSN12D = YES;

 RULE REPEAT_REQUEST_12D
```

```
IF
     MSN12_VALID <> UNKNOWN
THEN
     REPEAT_REQUEST2D = NO
ELSE
     REPEAT_REQUEST2D = YES
     CLS
     DISPLAY " POSITION 2 MUST BE N, OR O.
                  PRESS ENTER TO CONTINUE. ~"
     CLS
     RESET MSN1_2
     FIND MSN1_2;

RULE MISSION_POSITION_22A
IF
      MSN2_1 = 1 AND
      MSN22_VALID = UNKNOWN
THEN
        WHILETRUE MSN22_VALID = UNKNOWN THEN
           RESET CK_MSN22A
           FIND CK_MSN22A
           RESET REPEAT_REQUEST2A
           FIND REPEAT_REQUEST2A
        END
        MSN22_VALID = TRUE;

RULE USER_DEFINED_MISSION22A_CODE
IF
     MSN2_2 = A OR
     MSN2_2 = B OR
     MSN2_2 = C OR
     MSN2_2 = D OR
     MSN2_2 = E OR
     MSN2_2 = F OR
     MSN2_2 = G OR
     MSN2_2 = H OR
     MSN2_2 = I OR
     MSN2_2 = N OR
     MSN2_2 = O OR
     MSN2_2 = P OR
     MSN2_2 = R AND
     MSN2_1 = 1 AND
     EXCD <> X
THEN
     CK_MSN22A = YES
     MSN22_VALID = TRUE;

RULE USER_DEFINED_MISSION22AA_CODE
IF
     MSN2_2 <> A OR
     MSN2_2 <> B OR
```

```
      MSN2_2 <> C OR
      MSN2_2 <> D OR
      MSN2_2 <> E OR
      MSN2_2 <> F OR
      MSN2_2 <> G OR
      MSN2_2 <> H OR
      MSN2_2 <> I OR
      MSN2_2 <> N OR
      MSN2_2 <> O OR
      MSN2_2 <> P OR
      MSN2_2 <> Q AND
      MSN2_1 = 1 AND
      EXCD <> X
THEN
      CK_MSN22A = YES;

RULE REPEAT_REQUEST_2A
IF
      MSN22_VALID <> UNKNOWN
THEN
      REPEAT_REQUEST22A = NO
ELSE
      CK_MSN22A = YES
      REPEAT_REQUEST22A = YES
      CLS
      DISPLAY " POSITION 2 MUST BE R, A-I, OR N-P.
                    PRESS ENTER TO CONTINUE. ~"
      CLS
      RESET MSN2_2
      FIND MSN2_2;

RULE MISSION_POSITION_22B
IF
       MSN2_1 = 2 AND
       EXCD <> X AND
       MSN22_VALID = UNKNOWN
THEN
          WHILETRUE MSN22_VALID = UNKNOWN THEN
             RESET CK_MSN22B
             FIND CK_MSN22B
             RESET REPEAT_REQUEST22B
             FIND REPEAT_REQUEST22B
          END
          MSN22_VALID = TRUE;

RULE USER_DEFINED_MISSION22B_CODE
IF
      MSN2_2 = J OR
      MSN2_2 = K OR
      MSN2_2 = L OR
      MSN2_2 = M OR
```

```
      MSN2_2 = N OR
      MSN2_2 = O OR
      MSN2_2 = P OR
      MSN2_2 = Q OR
      MSN2_2 = R AND
      MSN2_1 = 2 AND
      EXCD <> X
THEN
      CK_MSN22B = YES
      MSN22_VALID = TRUE;

RULE USER_DEFINED_MISSION22BB_CODE
IF
      MSN2_2 <> J OR
      MSN2_2 <> K OR
      MSN2_2 <> L OR
      MSN2_2 <> M OR
      MSN2_2 <> N OR
      MSN2_2 <> O OR
      MSN2_2 <> P OR
      MSN2_2 <> Q OR
      MSN2_2 <> R AND
      MSN2_1 = 2 AND
      EXCD <> X
THEN
      CK_MSN22B = YES;

 RULE REPEAT_REQUEST_22B
IF
      MSN22_VALID <> UNKNOWN
THEN
      REPEAT_REQUEST22B = NO
ELSE
      REPEAT_REQUEST22B = YES
      CLS
      DISPLAY " POSITION 2 MUST BE IN THE RANGE OF J-R.
                    PRESS ENTER TO CONTINUE. ~"
      CLS
      RESET MSN2_2
      FIND MSN2_2;

RULE MISSION_POSITION_22C
IF
       MSN2_1 >= 3 AND
       EXCD <>  X AND
       MSN22_VALID = UNKNOWN
THEN
         WHILETRUE MSN22_VALID = UNKNOWN THEN
            RESET CK_MSN22C
            FIND CK_MSN22C
            RESET REPEAT_REQUEST22C
```

```
          FIND REPEAT_REQUEST22C
        END
        MSN22_VALID = TRUE;

RULE USER_DEFINED_MISSION22C_CODE
IF
     MSN2_2 = N OR
     MSN2_2 = O OR
     MSN2_2 = S OR
     MSN2_2 = T OR
     MSN2_2 = U OR
     MSN2_2 = V OR
     MSN2_2 = W OR
     MSN2_2 = X OR
     MSN2_2 = Y OR
     MSN2_2 = Z AND
     MSN2_1 >= 3 AND
     EXCD <> X
THEN
     CK_MSN22C = YES
     MSN22_VALID = TRUE;

RULE USER_DEFINED_MISSION22CC_CODE
IF
     MSN2_2 <> N OR
     MSN2_2 <> O OR
     MSN2_2 <> S OR
     MSN2_2 <> T OR
     MSN2_2 <> U OR
     MSN2_2 <> V OR
     MSN2_2 <> W OR
     MSN2_2 <> X OR
     MSN2_2 <> Y OR
     MSN2_2 <> Z AND
     MSN2_1 >= 3 AND
     EXCD <> X
THEN
     CK_MSN22C = YES;

 RULE REPEAT_REQUEST_22C
IF
     MSN22_VALID <> UNKNOWN
THEN
     REPEAT_REQUEST22C = NO
ELSE
     REPEAT_REQUEST22C = YES
     CLS
     DISPLAY " POSITION 2 MUST BE N, O, OR S-Z.
                  PRESS ENTER TO CONTINUE. ~"
     CLS
     RESET MSN2_2
```

```
     FIND MSN2_2;

RULE MISSION_POSITION_32A
IF
      MSN3_1 = 1 AND
      MSN32_VALID = UNKNOWN
THEN
        WHILETRUE MSN32_VALID = UNKNOWN THEN
           RESET CK_MSN32A
           FIND CK_MSN32A
           RESET REPEAT_REQUEST32A
           FIND REPEAT_REQUEST32A
        END
        MSN32_VALID = TRUE;

RULE USER_DEFINED_MISSION32A_CODE
IF
     MSN3_2 = A OR
     MSN3_2 = B OR
     MSN3_2 = C OR
     MSN3_2 = D OR
     MSN3_2 = E OR
     MSN3_2 = F OR
     MSN3_2 = G OR
     MSN3_2 = H OR
     MSN3_2 = I OR
     MSN3_2 = N OR
     MSN3_2 = O OR
     MSN3_2 = P OR
     MSN3_2 = R AND
     MSN3_1 = 1 AND
     EXCD <> X
THEN
     CK_MSN32A = YES
     MSN32_VALID = TRUE;

RULE USER_DEFINED_MISSION32AA_CODE
IF
     MSN3_2 <> A OR
     MSN3_2 <> B OR
     MSN3_2 <> C OR
     MSN3_2 <> D OR
     MSN3_2 <> E OR
     MSN3_2 <> F OR
     MSN3_2 <> G OR
     MSN3_2 <> H OR
     MSN3_2 <> I OR
     MSN3_2 <> N OR
     MSN3_2 <> O OR
     MSN3_2 <> P OR
     MSN3_2 <> R AND
```

```
     MSN3_1 = 1 AND
     EXCD <> X
THEN
     CK_MSN32A = YES;

RULE REPEAT_REQUEST_3A
IF
     MSN32_VALID <> UNKNOWN
THEN
     REPEAT_REQUEST32A = NO
ELSE
     CK_MSN32A = YES
     REPEAT_REQUEST32A = YES
     CLS
     DISPLAY " POSITION 2 MUST BE R, A-I, OR N-P.
                    PRESS ENTER TO CONTINUE. ~"
     CLS
     RESET MSN3_2
     FIND MSN3_2;

RULE MISSION_POSITION_3
IF
      MSN3_1 = 2 AND
      EXCD <> X AND
      MSN32_VALID = UNKNOWN
THEN
        WHILETRUE MSN32_VALID = UNKNOWN THEN
           RESET CK_MSN32B
           FIND CK_MSN32B
           RESET REPEAT_REQUEST32B
           FIND REPEAT_REQUEST32B
        END
        MSN32_VALID = TRUE;

RULE USER_DEFINED_MISSION32B_CODE
IF
     MSN3_2 = J OR
     MSN3_2 = K OR
     MSN3_2 = L OR
     MSN3_2 = M OR
     MSN3_2 = N OR
     MSN3_2 = O OR
     MSN3_2 = P OR
     MSN3_2 = Q OR
     MSN3_2 = R AND
     MSN3_1 = 2 AND
     EXCD <> X
THEN
     CK_MSN32B = YES
     MSN32_VALID = TRUE;
```

```
RULE USER_DEFINED_MISSION32BB_CODE
IF
     MSN3_2 <> J OR
     MSN3_2 <> K OR
     MSN3_2 <> L OR
     MSN3_2 <> M OR
     MSN3_2 <> N OR
     MSN3_2 <> O OR
     MSN3_2 <> P OR
     MSN3_2 <> Q OR
     MSN3_2 <> R AND
     MSN3_1 = 2 AND
     EXCD <> X
THEN
     CK_MSN32B = YES;

 RULE REPEAT_REQUEST_32B
IF
     MSN32_VALID <> UNKNOWN
THEN
     REPEAT_REQUEST32B = NO
ELSE
     REPEAT_REQUEST32B = YES
     CLS
     DISPLAY " POSITION 2 MUST BE IN THE RANGE OF J-R.
                    PRESS ENTER TO CONTINUE. ~"
     CLS
     RESET MSN3_2
     FIND MSN3_2;

RULE MISSION_POSITION_32C
IF
      MSN3_1 >= 3 AND
      EXCD <> X AND
      MSN32_VALID = UNKNOWN
THEN
        WHILETRUE MSN32_VALID = UNKNOWN THEN
           RESET CK_MSN32C
           FIND CK_MSN32C
           RESET REPEAT_REQUEST32C
           FIND REPEAT_REQUEST32C
        END
        MSN32_VALID = TRUE;

RULE USER_DEFINED_MISSION32C_CODE
IF
     MSN3_2 = N OR
     MSN3_2 = O OR
     MSN3_2 = S OR
     MSN3_2 = T OR
     MSN3_2 = U OR
```

```
      MSN3_2 = V OR
      MSN3_2 = W OR
      MSN3_2 = X OR
      MSN3_2 = Y OR
      MSN3_2 = Z AND
      MSN3_1 >= 3 AND
      EXCD <> X
THEN
      CK_MSN32C = YES
      MSN32_VALID = TRUE;

RULE USER_DEFINED_MISSION32CC_CODE
IF
      MSN3_2 <> N OR
      MSN3_2 <> O OR
      MSN3_2 <> S OR
      MSN3_2 <> T OR
      MSN3_2 <> U OR
      MSN3_2 <> V OR
      MSN3_2 <> W OR
      MSN3_2 <> X OR
      MSN3_2 <> Y OR
      MSN3_2 <> Z AND
      MSN3_1 >= 3 AND
      EXCD <> X
THEN
      CK_MSN32C = YES;

 RULE REPEAT_REQUEST_32C
IF
      MSN32_VALID <> UNKNOWN
THEN
      REPEAT_REQUEST32C = NO
ELSE
      REPEAT_REQUEST32C = YES
      CLS
      DISPLAY " POSITION 2 MUST BE N, O, OR S-Z.
                    PRESS ENTER TO CONTINUE. ~"
      CLS
      RESET MSN3_2
      FIND MSN3_2;


ASK WHICHTASK: "CHOOSE A TASK TO PERFORM ON THE DATABASE.";
CHOICES WHICHTASK: APPEND_RECORD, UPDATE_RECORD,
DELETE_RECORD, DISPLAY_RECORD, EXIT;
ASK DOCNUM_NEW: " ENTER THE NEW DOCUMENT NUMBER.";
ASK SIDENUM_NEW: " PLEASE INDICATE THE SIDE NUMBER OF THE
AIRCRAFT.";
ASK SIDENUM_UPDATE: " PLEASE INDICATE THE NEW SIDE NUMBER.";
```

```
ASK EXCD: " ENTER AN EXCEPTION CODE OR <SPACE> FOR NONE.";
ASK MSN1_1: " ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH
POSITION ENTRY.";
ASK MSN1_2: " ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH
POSITION ENTRY                {MSN1_1}   ENTER SECOND
POSITION";
ASK MSN1_3: " ENTER A MISSION 1 CODE, HIT ENTER AFTER EACH
POSITION ENTRY                {MSN1_1}{MSN1_2}   ENTER THIRD
POSITION";
ASK HRS1: " ENTER THE HOURS FLOWN ON MISSION 1.";
ASK MSN2_1: " ENTER A MISSION 2 CODE, HIT ENTER AFTER EACH
POSITION ENTRY";
ASK MSN2_2: " ENTER A MISSION 2 CODE, HIT ENTER AFTER EACH
POSITION ENTRY                {MSN2_1}   ENTER SECOND
POSITION";
ASK MSN2_3: " ENTER A MISSION 2 CODE, HIT ENTER AFTER EACH
POSITION ENTRY                {MSN2_1}{MSN2_2}   ENTER THIRD
POSITION";
ASK HRS2: " ENTER THE HOURS FLOWN ON MISSION 2.";
ASK MSN3_1: " ENTER A MISSION 3 CODE, HIT ENTER AFTER EACH
POSITION ENTRY                {MSN3_1}{MSN3_2}{MSN3_3}";
ASK MSN3_2: " ENTER A MISSION 3 CODE, HIT ENTER AFTER EACH
POSITION ENTRY                {MSN3_1}   ENTER SECOND
POSITION";
ASK MSN3_3: " ENTER A MISSION 3 CODE, HIT ENTER AFTER EACH
POSITION ENTRY                {MSN3_1}{MSN3_2}   ENTER THIRD
POSITION";
ASK HRS3: " ENTER THE HOURS FLOWN ON MISSION 3.";
ASK TOTFLT: " ENTER THE TOTAL NUMBER OF FLIGHTS.";
ASK OPS: " ENTER THE SHIP/FIELD OPERATIONS CODE.";
ASK CJ: " ENTER THE NUMBER OF CATAPULT SHOTS OR JATO
LAUNCHES.";
ASK NUMHOISTS: " ENTER THE NUMBER OF AIRCRAFT HOISTS.";
ASK ENGHRS: " ENTER HOURS FOR ENGINE {ENGNUM}.";
ASK UPDATE_ENGHRS: " ENTER HOURS FOR ENGINE
{ENGINE_NUMBER}.";
ASK DOCNUM_VIEW: " WHICH NAVAL AIRCRAFT FLIGHT RECORD DO YOU
WANT TO VIEW.";
ASK DOCNUM_DELETE: " WHICH NAVAL AIRCRAFT FLIGHT RECORD DO
YOU WANT TO DELETE.";
ASK DOCNUM_UPDATE: " WHICH NAVAL AIRCRAFT FLIGHT RECORD DO
YOU WANT TO UPDATE.";
ASK FIELD_TO_UPDATE: " SELECT WHICH FIELD YOU WANT TO
UPDATE.";
CHOICES FIELD_TO_UPDATE: DOCUMENT_NUMBER, SIDE_NUMBER,
EXCEPTION_CODE, MISSION_CODE_1, MISSION_1_HOURS,
MISSION_CODE_2, MISSION_2_HOURS, MISSION_CODE_3,
MISSION_3_HOURS, TOTAL_FLIGHTS, SHIP_FIELD_OPERATIONS_CODE,
CATAPULT_JATO_LAUNCHES, AIRLIFT_MISSION_NUMBER,
NUMBER_OF_HOISTS, ENGINE_HOURS, DONE;
ASK EXCD_NEW: " ENTER AN EXCEPTION CODE OR <SPACE> FOR
```

```
NONE.";
ASK ENGINE_NUMBER: " CHOOSE THE ENGINE NUMBER THAT YOU WANT
TO CHANGE THE HOURS FLOWN.";
ASK CONTINUE: " THIS ACTION WILL DELETE THE WHOLE FLIGHT
RECORD! DO YOU WANT TO CONTINUE?";
CHOICES CONTINUE: YES, NO;
```

LIST OF REFERENCES

1. Fernandez,E.B., Summers, R.C.,and Wood,C., *Database Security and Integrity*, Addison-Wesly Publishing Company Inc., 1981.

2. Codd,E.F., *The Relational Model for Database Management*, Version 2, Addison-Wesly Publishing Company Inc., 1990.

3. Department of the Navy, Office of the Chief of Naval Operations, OPNAVINST 3710.7N, *Natops General Flight and Operating Instructions*, 10 April 1990.

4. Kroenke,D.M., and Dolan,K.A., *Database Processing*, 3rd ed., Science Research Associates Inc., Chicago, Illinois, 1988.

5. Shafer,S.L., and Westney,R.E., "Six Steps To Successful Expert Systems,"*Cost Engineering*, v.30, p.17, June 1988.

6. Department of the Navy. Office of the Chief of Naval Operations, OPNAVINST 4790.2E, *The Naval Aviation Maintenance Program*, 1 January 1989.

BIBLIOGRAPHY

Guida,G., and Tasso, C., *Topics in Expert System Design*, Elsevier Science Publishers B.V., North Holland, 1989.

Kerschberg, L., *Expert System Databases*, The Benjamin/Cummings Publishing Company, Menlo Park, California, 1987.

Rolston, D.W., *Principles of Artificial Intelligence and Expert Systems Development*, McGraw-Hill Inc., 1988.

Moose, A., Schussler, T., and Shafer, D., *VP-Expert*, Paperback Software International, Berkeley, California, 1989.

Whitten, J.L., Bentley, L.D., and Ho, T.L., *Systems Analysis and Design Methods*, Times Mirror/Mosby College Publishing, St. Louis, Missouri, 1986.

INITIAL DISTRIBUTION LIST

		NO. COPIES
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Dudley Knox Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Prof. Magdi N. Kamel, Code AS/KA Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	2
4.	Prof. Hemant K. Bhargava, Code AS/BH Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
5.	Commanding Officer Naval Sea Logistics Center Code 612.2 5450 Carlisle Pike P.O. Box 2060 Mechanicsburg, Pennsylvania 17055-0795	1
6.	LT. George J. Salitsky, USN 117 School St. Childs, Pennsylvania 18407	2